A Security Policy Framework for Grid Services

A Thesis submitted

for the award of the degree of

DOCTOR OF PHILOSOPHY

By

Sarbjeet Singh

(9040355)

under the guidance of

Dr. Seema Bawa

Professor & Head

Department of Computer Science and Engineering

Thapar University, Patiala, Punjab – 147 004

Department of Computer Science and Engineering

Thapar University, Patiala – 147 004, INDIA

JULY 2008

Table of Contents

List of Figures

List of Tables

Certificate

I hereby certify that the work which is being presented in this thesis entitled **"A Security Policy Framework for Grid Services"**, in partial fulfillment of the requirement for the award of degree of "Doctor of Philosophy" submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Seema Bawa and refers other researchers works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

Sarbjeet Singh 16/06/09

(Sarbjeet Singh)
Regn. No. 9040355

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Seema Bawa 16/06/2009

(Dr. Seema Bawa)
Professor & Head
Department of Computer Science & Engineering
Thapar University
Patiala – 147 004
Punjab, INDIA

Acknowledgements

I would like to express my sincere gratitude to Professor Seema Bawa for her valuable advices, direction, and encouragement during my doctoral research endeavor for the past four years. Her observations and comments helped me to remain focused and move forward in the direction of improvement. It was only because of her support and all around help that I am able to complete the thesis in time. I thank her for providing me the opportunity to work with her.

I would like to express my sincere thanks to Doctoral Committee Members – Professor R.S. Kaler and Professor R.K. Sharma, and Dean of Research and Sponsored Projects – Professor Sushil Mittal, for their critical observations and valuable comments which helped enormously in presenting results and shaping this thesis.

I am thankful to Dr. Maninder Singh, Assistant Professor, for their valuable suggestions and help in setting up the environment for framework implementation.

I would like to thank faculty and staff members of Computer Science & Engineering of Thapar University, Patiala for providing and sharing resources that have been utilized in different implementations of the framework.

I am also grateful to colleagues and staff members of University Institute of Engineering and Technology, Panjab University, Chandigarh for their cooperation. They always helped me to spare time for research.

I would like to express special thanks to Anju Sharma (Thapar University), Inderveer Channa (Thapar University) and Bhupinder Singh (Siemens) for making research a wonderful experience for me.

I would like to thank administrative and academic staff members of Thapar University and Panjab University who have been kind enough to advise and help in their respective roles.

I am also grateful to numerous others who have directly or indirectly contributed towards carrying out the research in all aspects during last four years.

Last, but not the least, I would like to dedicate this thesis to my family, my father, my mother, my wife and twin daughters Agampreet and Avneet for their love, patience and understanding. They allowed me to spend most of the time on this thesis.

Sarbjeet Singh 16/06/09

Sarbjeet Singh

Abstract

Grid computing deals with flexible, secure and coordinated sharing of resources that are distributed over wide area networks. With the evolution of this field, the complexity of the distributed systems has increased and therefore the implementation of a secure environment has become difficult. At the same time, grid setups necessarily require a secure environment where users/organizations have access to resources, precisely on the basis of their rights, with proper accountability and control. This thesis work implements a security policy framework to address key security requirements (mainly identified as authentication, privacy, trust and authorization) and provide support to express, evaluate and enforce security policies related to these requirements.

The identified security requirements of grid systems have been categorized mainly into four security disciplines which are authentication, privacy, trust and authorization. Therefore, the framework implements four different models namely authentication model, privacy model, trust model and policy based authorization model. These models address security requirements and policies specific to their respective disciplines.

To achieve the set objectives, a comprehensive literature review of developments related to grid and web services, their method of operation and execution has been done. The similarities and differences between the two have been brought out. A thorough study and analysis of standards and specifications used in grid and web services based systems has also been carried out. Previous work done in the areas of authentication, privacy, trust and policy based authorization in grid systems has been studied, extended in the form of a framework, and reported in detail.

Out of the four models, the authentication model provides support for single sign-on and delegation features using proxy certificates and a credential management service to store, retrieve and update multiple user credentials. The privacy and trust models provide privacy and trust based access to grid services. The privacy model in particular provides support for anonymous access, hidden service access and access to private information based on conformance to privacy policies. The trust model provides support for

calculating direct as well as recommended trust to determine trustworthiness of target services/resources. All these models also describe how the security policies related to them can be expressed and evaluated. The policy based authorization model provides access to grid services based on conformance to various types of security policies. The policy specification, evaluation and enforcement related functionality of authentication, privacy and trust models has been incorporated into policy based authorization model and the resulting model is called the integrated policy based authorization model.

The complete framework has been evaluated by implementing different security related scenarios and through implementations involving enforcement of different types of access control policies. These scenarios and implementations cover different aspects related to authentication, privacy, trust and authorization. The results show that the various implementations are able to meet the identified security requirements. The results clearly demonstrate that the approach is workable and can be effectively used to address key security requirements related to authentication, privacy, trust and authorization, and further to provide policy based access to grid services/resources.

Chapter 1

Introduction

This chapter provides a high level overview of grid computing, grid and web services, similarity and differences among them, their architecture, the standards and specifications available and used in designing middleware for them. The chapter briefly discusses the role of XML, SOAP, WSDL and UDDI in grid systems. It also presents overview of web services security related specifications like WS-Security, WS-Trust, WS-SecureConversation, XACML and SAML *etc.* The chapter ends with a discussion on the organization of the rest of the thesis.

1.1 Distributed Computing

Computer networks are being widely used these days by almost every industry, academic, research and government organization. These networks facilitate the sharing of hardware and software resources and allow immediate transfer of information over distances ranging from less than a meter to thousands of kilometers. Such an arrangement also allows jobs and processes to be distributed to separate computers where they can be executed in parallel resulting in an increase in number of jobs in unit time. This concept led to the birth of distributed computing. In a distributed system, the problem is divided into tasks that can be distributed among several computers on a network and executed in parallel. Distributed computing is a programming model in which processing occurs at many different places (or nodes) around a network. A distributed system is a set of interconnected, autonomous computers that cooperatively solve large, single problem by facilitating parallel execution of separate but possibly related tasks [1]. Parallelism can either be data parallelism or functional parallelism. In data parallelism each computer performs the same function but on different sets of data and in functional parallelism each computer performs different function on same or different sets of data.

Today, several types of distributed computing systems exist like peer-to-peer computing, cluster computing, utility computing, autonomic computing, pervasive computing, grid computing, *etc.* Many of these share common goals and concerns but differences among them are in terms of their focus, environment and mechanisms used for their implementation. Grid Computing is a type of distributed computing that focuses on large scale sharing of geographically dispersed resources. Following section defines grid computing, the concept of virtual organizations and application areas of grid computing.

1.2 Grid Computing

The term "the Grid" was coined in mid 1990s to describe a large scale distributed computing infrastructure for advanced science and engineering. Since then, it has emerged as an important and interesting field that is different from conventional distributed computing by its focus on large scale resource sharing, innovative applications, and, in some cases, high performance orientation [2], [3]. Grid computing resources include computing power, data storage, hardware instruments, on-demand software and applications *etc.* [2], [3], [4], [5]. Grid computing can be thought of as an enhanced or extended form of distributed computing. It deals with flexible, secure and coordinated sharing of resources that are distributed over wide area networks. Grid concept is defined as the controlled and coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations [4], [6]. As the focus of grid is on dynamic, cross-organizational sharing, grid technologies complement rather than compete with existing distributed computing technologies [2].

The concept of virtual organization is central to grid computing. A Virtual Organization (VO) is a dynamic grouping of individuals, multiple groups, physical organizations or administrative domains defined around a set of resource sharing rules and conditions like what is shared, who is allowed to share and the conditions under which sharing takes place [6], [7], [8], [9]. The VOs share some commonality among them, like common concerns and requirements, but may vary in their size, structure, purpose, scope and community. The sharing of VO resources is accomplished within controlled and well defined conditions and policies.

With the evolution of this field, the complexity of the distributed systems has increased and the implementation of a secure environment has become difficult. Grid systems require unique methods for resource management, scheduling, discovery and security. There are a number of activities addressing these issues and a lot of research is going on in these areas [4]. This thesis focuses on grid security area with the aim to implement a security policy framework to enable secure access to grid services/resources.

1.2.1 Grid Application Areas

Grids are typically used for solving large scale resource sharing and compute intensive problems. Grid application areas include compute-intensive, data-intensive, sensor-intensive, knowledge-intensive, and collaboration-intensive scenarios and environment. A knowledge-intensive environment provides knowledge-intensive support to business, scientific or other processes and consists of infrastructure and services which are reliant on professional or expert knowledge in particular area(s). A collaboration-intensive environment consists of multiple systems/units working in collaboration for the solution of business, scientific or other problems. These areas address problems ranging from multiplayer video gaming, fault diagnosis in jet engines, and earthquake engineering to bioinformatics, biomedical imaging and astrophysics *etc.* [10]. Grid concepts can also be used to solve problems like VLSI Test Generation [11] and distributed system level diagnosis [12]. Grid security technology can also complement applications like Worldwide Computing Middleware [13]. Some of the projects representing broad spectrum of grid application areas include Distributed Aircraft Maintenance Environment (DAME), Network for Earthquake Engineering Simulation (NEES), World Wide Telescope, Earth System Grid (ESG), Biomedical Informatics Research Network (BIRN), Grid Physics Network (GriPhyN), TeraGrid, European Union Data Grid Project *etc.* [10].

Many of the organizations have started identifying the major grid computing business areas also. These include financial services (for running long, complex financial models for arriving at more accurate decisions), higher education (for enabling advanced data and computation intensive research), engineering services (for collaborative design and data intensive testing), life sciences (for analyzing and decoding strings of biological and chemical information), government (for enabling seamless collaboration and agility in both civil and military departments) *etc.*[6]. Grid computing enables organizations (real

and virtual) to take advantage of various resources in ways not previously possible. Organizations can take advantage of underutilized resources to meet business requirements while minimizing additional costs [5]. In future we are expecting their presence in almost every field.

1.3 Grid and Web Architecture

The terms “Grid” and “Web” both represent distributed systems. Just as the web revolutionized information sharing by providing a universal protocol and syntax (HTTP and HTML) for information exchange, the grid tries to standardize general resource sharing by proposing standard protocols and syntaxes. Though they started far apart in standards, specifications, technology, scope and other concerns, but now they are merging into a common platform. Following paragraphs briefly discuss the individual grid and web services architecture.

1.3.1 Grid Architecture

The most notable architecture for grid systems is proposed by Ian Foster *et al.* [2]. The goal of the architecture is to identify the basic components of a grid system, the purpose and function of components and their interaction among each other. The main attention of the architecture is on interoperability among resource providers and users to establish sharing relationships. The architecture organizes components into layers. The components within each layer share common characteristics but can build on capabilities and behaviors provided by any lower layer [2]. Grid protocol architecture consists of five layers: Fabric, Connectivity, Resource, Collective and Application as shown in Figure 1.1. This architectural description is high level and places few constraints on design and implementation.

The grid fabric layer provides resources to which shared access is mediated by grid protocols. Fabric components implement the local, resource specific operations that occur on specific resources as a result of sharing operations at higher levels.

The connectivity layer defines core communication and authentication protocols required for grid specific network transactions. Communication protocols enable the

exchange of data between fabric layer resources. Authentication protocols build on communication services provide cryptographically secure mechanisms for verifying the identity of users and resources.

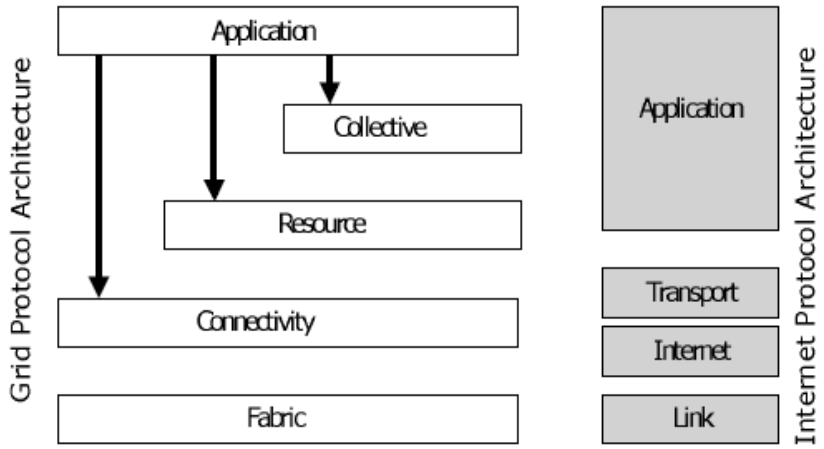

Figure 1.1: The layered Grid architecture and its relationship to the Internet protocol architecture [2]

Resource layer builds on connectivity layer communication and authentication protocols to define protocols for the secure negotiation, initiation, monitoring, control, accounting, and payment of sharing operations on individual resources. Resource layer implementations of these protocols call fabric layer functions to access and control local resources.

Collective layer contains protocols and services that are not associated with any one specific resource but rather are global in nature and capture interactions across collections of resources. Collective functions can be implemented as persistent services with associated protocols.

Application layer comprises the user applications that operate within a VO environment. Applications are constructed in terms of, and by calling upon, services defined at any layer. At each layer there are well defined protocols that provide access to useful services [2].

1.3.2 Web Services Architecture

The Web Services Architecture [14] is an interoperability architecture. This is a standard initiative from W3C [15]. Web Services Architecture provides a conceptual model and a

context for understanding web services and other relationships between the components of this model. The architecture has four models namely message oriented model (focusing on messages, message structure, message transport and so on), service oriented model (focusing on aspects of service, action and so on), resource oriented model (focusing on resources present in the environment) and policy model (focusing on constraints on the behavior of agents and services) [14]. For constructing interoperable web services, the architecture attempts to define and relate important technologies like XML, SOAP and WSDL *etc.* Figure 1.2 provides illustration of some of these technologies.

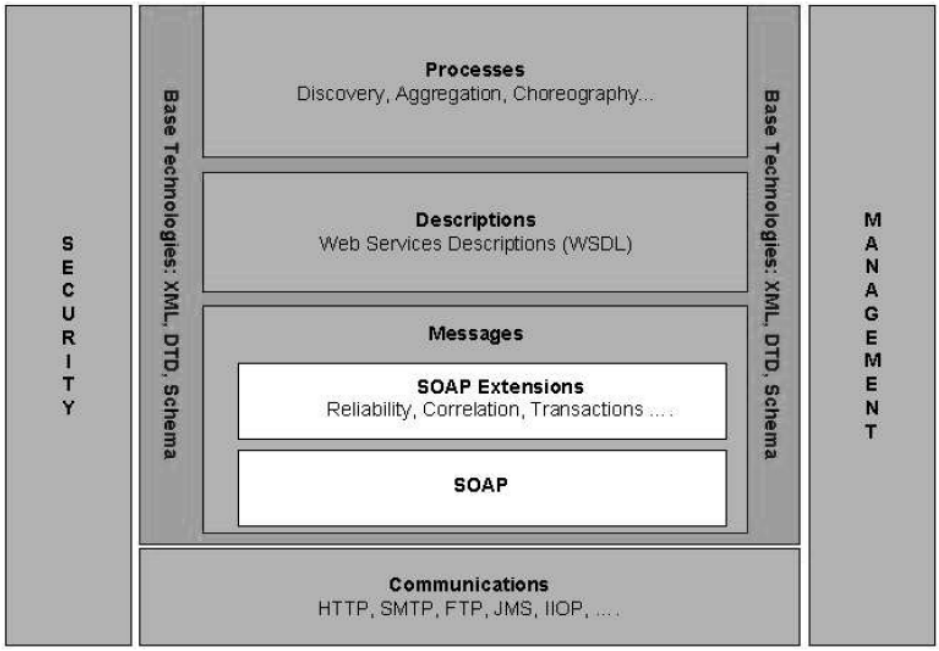

Figure 1.2: Web Services Architecture Stack [14]

The Web Service Architecture is built around XML technologies. It is independent of the underlying transport mechanisms. The messages exchanged between requesters and services forms the base layer of this architecture. These messages can be packaged and exchanged using SOAP and its extension models. SOAP provides a standard, extensible and composable framework for packaging and exchanging XML messages. The SOAP extension models provide a number of SOAP header messages for message correlation, transactional capabilities, message reliability and service addressing. A high-level description on the messages exchange and interaction pattern is also there. This description can be given through any description language of choice. The most notable among these description languages is the Web Services Description Language (WSDL).

A number of technologies can be build around this architectural model. These technologies can be high-end applications, infrastructure software and middleware solutions. The other notable features of the architecture are the vertical pillars for security and management, which are needed for all the horizontal architecture components. Web services architecture is a key enabler of the overall computing discipline of grid computing [6].

1.4 Grid and Web Services

A web service is defined by W3C [15] as a software system designed to support interoperable machine-to-machine interaction over a network. A web service is identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems.

A grid service instance is a web service that conforms to a set of conventions expressed by WSDL as service interfaces, extensions and behaviors. A grid service provides the controlled management of the distributed and often long lived state that is commonly required in sophisticated distributed applications [6]. Following sections briefly discuss important grid technological viewpoints and relationship between grid and web services.

1.4.1 OGSA, OGSI and WSRF

The Open Grid Services Architecture (OGSA) [3] developed by Global Grid Forum [16] aims to define a common, standard and open architecture for grid based applications. The goal of OGSA is to standardize practically all the services that one commonly finds in a grid application by specifying a set of standard interfaces for these services [17]. OGSA enables the creation, maintenance and integration of grid services by virtual organizations. OGSA is based on several other web services technologies like WSDL and SOAP *etc.* but aims to be largely agnostic in relation to the transport-level hiding of data.

OGSI (Open Grid Services Infrastructure) [18] was a core component of OGSA which provides a uniform way to describe grid services and defines a common pattern of behavior for all grid services. More specifically, the OGSI defines mechanisms for

creating, managing and exchanging information among grid services [5]. Based on the OGSI specification, a grid service instance is a web service that conforms to a set of conventions expressed by the WSDL as service interfaces, extensions and behaviors. According to the definition of OGSI, every grid service is a web service but the converse need not be true [6]. OGSI is now obsolete and has been superseded by WSRF.

Web Services Resource Framework (WSRF) [19] is a specification developed by OASIS [20] that specifies how web services can be made stateful. WSRF defines a set of specifications for defining the relationship between web services (that are normally stateless) and stateful resources. WSRF is a joint effort by the grid and web services communities [21].

1.4.2 Relationship between Grid and Web services

The difference between a grid and a web service lies in how the state is managed. Generally, service state management is classified into two forms: Interaction aware state and Application aware state [6]. In interaction aware state, interactions are correlated using some information passed from the client to the service, along with the message. This can be a simple cookie, a session ID, or complex correlated information. The advantage of this architecture design is that the service side is not managing any specific client state information, or creating a specific instance for a client. The server side implementation is very scalable and stateless in nature. On the other hand, in application aware state, services are aware of its client and create a specific instance for the specific client, and pass that instance information back to the client for interaction. The client is holding a reference to the specific instance of the service/application, and hence, can interact with the service instance without passing any correlation information. These services are typically referred to as stateful services because the state information is held in the service itself and not passed back to the client [6]. Grid services are stateful web services with a well defined set of interfaces and behaviors for interaction. Figure 1.3 illustrates the difference between a stateless web service, a stateful web service and a grid service.

Figure 1.3 (a) shows a stateless web service where two clients are accessing the same instance of web service A. Figure 1.3 (b) represents a stateful web service where each client is accessing its own instance of web service and each instance is capable of

maintaining state. Figure 1.3 (c) represents a grid service which is also stateful but defined according to OGSI specification.

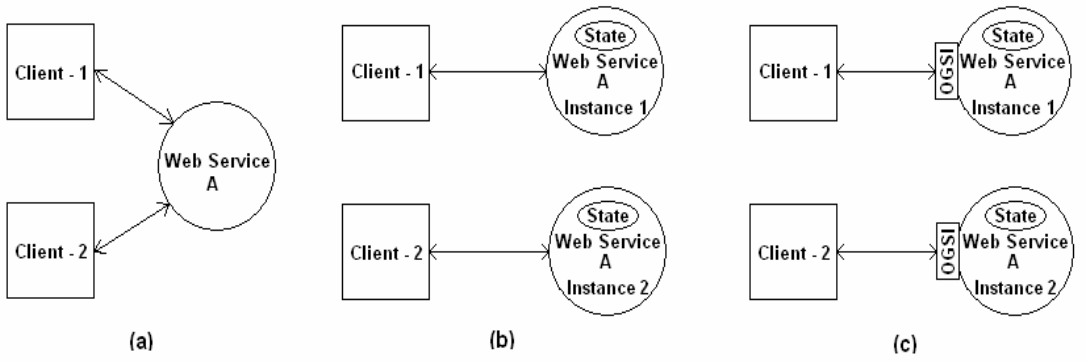

Figure 1.3: Schematic showing (a) Stateless web service (b) Stateful web service (c) Grid service.

As already mentioned that OGSI has been superseded by WSRF, nowadays WSRF defines how web services can be made stateful. Many of the security requirements of grid services overlap deeply with the security requirements of web services as grid services are stateful web services.

1.4.3 Grid and Web Services Invocation

The most important improvement of grid service over web service is its ability to maintain state. Grid services are stateful and potentially transient services. A grid service instance created at server side can be shared by two clients and one client can also have access to more than one instances. These instances are transient as they have a limited lifetime which is not bound to the lifetime of the grid service container. Web services are transient as their lifetime is bound to the web services container. Grid service can be persistent also, just like a normal web service. Choosing between persistent grid service and non persistent grid service depends entirely on the requirements of the application at hand [22].

Figure 1.4 illustrates how a web service is invoked. A brief description of the steps shown in Figure 1.4 is also given.

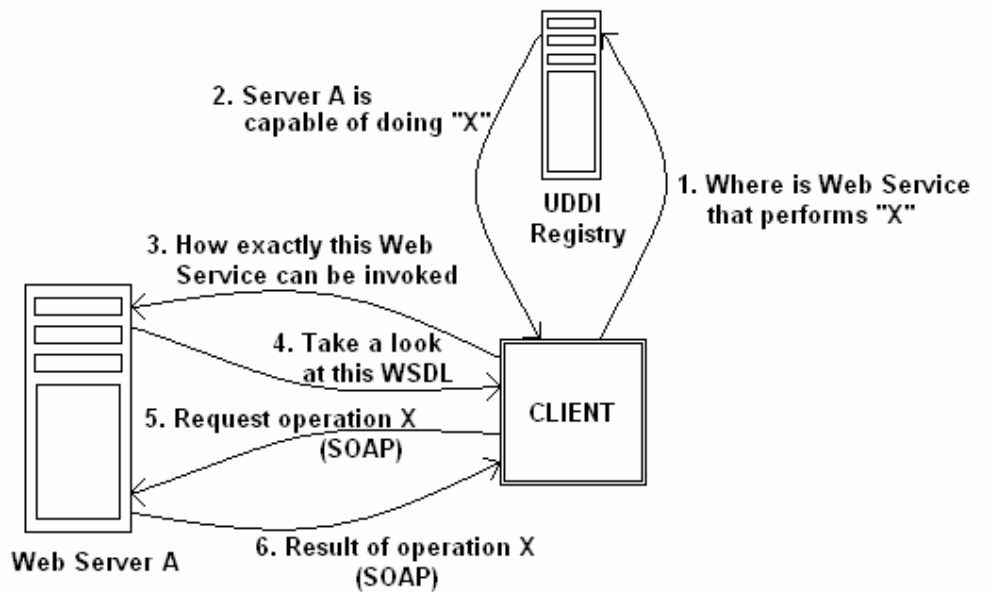

Figure 1.4: A typical web service invocation scenario

Step 1: Client contacts UDDI registry to find web service that performs task "X". This step is required as client may have no knowledge of the web service that performs task "X". UDDI registry is a global registry that has information about all the registered web services.

Step 2: UDDI response by returning information about the location of the web service that performs this task.

Step 3: With this information, client contacts web server "A" to know how exactly this service can be invoked.

Step 4: The web server replies in a language called WSDL. WSDL has all the necessary information required to invoke the service. WSDL is explained in Section 1.6.1 also.

Step 5: Client sends SOAP request to invoke the service.

Step 6: The web service returns the result of operation in SOAP.

The similar steps can be followed to invoke grid services also. The differences can be in terms of software used for implementing discovery services or grid service containers but the basic idea behind each step in grid service invocation is same as that for web service invocation. The main difference between a grid and a web service lies in how the service state is managed at server end and not in the basic method used for their invocation.

1.5 Grid Security

The real power of grid can be harnessed only if it provides secure access to resources/services. Grid security is a multidimensional problem [23]. A host of security issues need to be addressed to gain wide acceptance. Grid setups require a secure environment where users have access to resources precisely on the basis of their rights and where the system has record of their activities which they later cannot deny. The traditional approaches to address security requirements in distributed systems do not work for grid systems as grids present entirely new security issues and concerns. The dynamic and multi institutional nature of grid environment introduces several challenging security issues that require new technical approaches [24]. In addition to providing basic security requirements like authentication, authorization, confidentiality and integrity, a grid security infrastructure must be able to support more advanced security features like dynamic delegation of access rights, single sign-on/sign-off, multiple credentials management, dynamic establishment of trust relationships among participating domains, privacy, policy and trust related security issues in federated environments *etc.* [25]. In the field of cryptography, a lot of work has already been done to ensure confidentiality, integrity, secure conversation and nonrepudiation requirements using encryption and signature techniques that involve the use of symmetric as well as asymmetric keys. These existing techniques can be used to address similar requirements in grid systems also but the other security requirements like single sign-on, delegation, credential management, privacy and trust related issues in federated environments *etc.* require new approaches. There are several factors that make security hard *e.g.* user population and resource pool is large and dynamic, resources have different authentication and authorization requirements, computations span over multiple domains, users have different roles/privileges in different domains *etc.* [24]. All these factors make security a big and challenging issue.

Many of the unique security problems in grid arise because of the geographically and organizationally distributed nature of grids. A Grid offers uniform access to resources that are distributed geographically and organizationally. These different organizations can have radically different security policies that must be reconciled to allow for the coordinated usage of resources [10]. The heterogeneous nature of resources and their differing security policies make security implementation complicated and complex. Grid

also differs from other environments, in that most applications require the coordinated usage of multiple resources at multiple sites. The execution of a grid job may span over multiple sites. A grid job can also take a long time to execute ranging from few hours to days, weeks or even months. The authentication requirements at different sites can also be different. These characteristics require basic authentication mechanisms to be extended to include features like single sign-on, delegation, credential management and the procedures to integrate different authentication mechanisms.

In grid, user base is large and the access rights of a user are different in different organizations which are part of the grid. So mechanisms to manage different users' access rights in different organizations are also required. The authorization issue becomes more complex given the fact that organizations providing resources in a grid are free to use any authorization mechanism of their choice. There is no restriction to use a common authorization mechanism. The access policies of resources offered by different organizations can also be different. The resource/services can have different authentication, privacy, trust and other security policies associated with them. So mechanisms to express, evaluate and enforce these policies should also be there. Different organizations can also have different privacy and trust relationships among them. These relationships and policies play an important role in determining access to resources/services exposed in a grid [26]. Like authentication, procedures to integrate different authorization mechanisms are also required. Interoperability is another important feature that is essentially required for security solutions to be widely used and acceptable.

All the issues and scenarios described above require extension of basic security mechanisms. The main challenge in grid security is to establish security relationships not simply between a client and a server, but among many of the entities involved in the solution of a particular problem and to enable it to support a spectrum of security requirements [23].

1.6 Building Blocks for Grid Security

The concept of grid and web services is build around technologies like XML, SOAP, WSDL and UDDI *etc.* A number of web services security specifications like WS-Security, WS-SecureConversation, WS-Trust, WS-Policy, XACML and SAML *etc.* exist

to address different security related concerns. This section briefly discusses each of these technologies and specifications.

1.6.1 XML, SOAP, WSDL and UDDI

Although XML [27] stands for eXtensible Markup Language, the acronym "XML" is mostly used to describe not only XML itself, but also the ever-growing family of related technologies. It is designed to describe data and to improve the functionality of web by providing more flexible and adaptable ways of representing information. Since its introduction, XML (eXtensible Markup Language) [27] has revolutionized the way the information is structured, described and exchanged [28]. For web and grid services, the importance of XML is paramount as all key web and grid services technologies are based on it. XML is a specification, defined by W3C (World Wide Web Consortium) which defines a syntax used to define markup languages. XML defines syntax to structure a document using markup. All type of interoperable information regarding grid and web services is expressed in XML.

SOAP (Simple Object Access Protocol) [29] is a W3C specification and an XML based protocol for exchange of information in a decentralized, distributed environment. SOAP enables communication between web services. It provides a standard method for encoding data into a portable format but makes no distinction about how that data is to be transported from application to application [30]. It can bind to arbitrary underlying transport protocols such as HTTP, IIOP *etc*. The main benefit of SOAP is that it is lightweight protocol. It does not require enormous amount of work on part of either the sender or the recipient to communicate using the protocol [30]. A SOAP message consists of required Envelope and Body elements and optional Header and Fault elements. Envelope element identifies the XML document as a SOAP message, Header element contains header information and Body element contains call and response information. Fault element provides information about the errors that can occur while processing the message.

WSDL (Web Services Description Language) [31] is an XML language that provides a model and XML format for describing web services. WSDL provides facilities for service developers to separate abstract definitions of the service (interface, message and schema definitions) from concrete definitions (service, port and binding definitions).

WSDL works in conjunction with SOAP and UDDI to enable web services to interact with other web services, applications and devices across the internet. Actually, UDDI provides the ability to publish and locate a web service, WSDL describes the web service and SOAP provides ability to send messages between web services along with transport information. WSDL is a W3C submitted specification supported by a number of industry leaders including Microsoft and IBM. WSDL document consists of the following elements:

Types: A container for data type definitions.

Message: A definition of the data being communicated.

Operation: A description of an action supported by service.

Port Type: A set of operations supported by one or more endpoints.

Binding: A concrete protocol and data format specification for a particular port type.

Port: A single endpoint defined as a combination of a binding and the network address where it can be found.

Service: A collection of related endpoints.

UDDI (Universal Description, Discovery and Integration) [32] provides a method for publishing and finding service descriptions. It is a directory where web services can be registered and assigned to service providers. It enables business to quickly, easily and dynamically find businesses and transact with each other. There are many directory services, both internet and intranet, based on UDDI, which can be treated as global registries for web services. It is a cross-industry initiative to create a global registry of web services. The specifications are maintained by the UDDI organization. We can publish our web service to UDDI and search for other web services in it. By exposing the UDDI directory through SOAP, we can find web services at runtime [30].

1.6.2 Web Services Security Specifications

In April 2002, IBM and Microsoft released their joint "Security in a Web Services World: A Proposed Architecture and Roadmap" document [33] which defined a security framework for web services. The framework consists of different security related specifications. WS-Security [34] was the first specification which was released. Later, the specifications WS-Trust [35], WS-Policy [36], WS-SecureConversation [37] and WS-Federation [38] were released.

WS-Security [34] describes how signature and encryption headers are attached to SOAP messages. In addition, it also describes how to attach security tokens, including binary security tokens such as X.509 and Kerberos tickets, to messages. WS-Policy [36] provides a general purpose model and corresponding syntax to describe the policies of a web service, *e.g.* required security tokens, supported encryption algorithms and privacy rules *etc.* WS-Trust [35] describes a framework for trust models that enables web services to securely interoperate. It provides a framework for requesting and issuing security tokens. WS-Privacy is not available as of June 2008, but will describe a model for how web services and requesters can state subject privacy preferences and organizational privacy practice statements [33]. WS-SecureConversation [37] describes how to manage and authenticate message exchanges between parties including security context exchange and establishing and deriving session keys. WS-Federation [38] describes how to manage and broker the trust relationships in a heterogeneous federated environment including support for federated identities. WS-Authorization is not available as of June 2008, but will describe how to manage authorization data and authorization policies [33].

1.6.3 Security Assertion Markup Language (SAML)

The OASIS SAML [39] specification allows trust assertions to be specified using XML. These assertions can concern authorizations, authentications and attributes of specific entities. An assertion can be defined as a claim, statement or declaration. In addition to assertions, the SAML specification also defines a client/server protocol for exchanging XML message requests and responses. SAML enables “portable trust” by supporting the assertion of authentication of single principals between different (and potentially multiple) domains [40]. SAML provides three types of assertions: authentication assertion, authorization decision assertion and attribute assertion. Each type of assertion can have its own policies, profiles and attributes. Following paragraphs explain each of these assertions.

Authentication Assertion: The authentication authority receives a set of credentials from the credentials collector, and processes them according to a specific policy. An authentication assertion can then be made with respect to the two other assertions (authorization decision assertion and attribute assertion), if the authority determines that the credentials are valid. The assertion defines several authentication elements such as the

integrity of the issuer and the principal, the time the authentication was granted, and how long the authentication is valid. The assertion can clearly indicate that a principal was authenticated by a specific system at a specific point in time.

Authorization Decision Assertion: An authorization decision assertion involves making a decision about whether or not a principal can access a specific resource, given an authentication assertion and an attribute assertion. In involves two entities: Policy Decision Point (PDP) and the Policy Enforcement Point (PEP). According to a policy, the PDP and PEP make and enforce authorization decisions respectively.

Attribute Assertion: An attribute assertion begins with an attribute authority accepting an authentication assertion and using a policy to determine the privileges of a principal. The attribute assertion can be passed to a Policy Decision Point (PDP) for authorization.

1.6.4 eXtensible Access Control Markup Language (XACML)

XACML [41] is an initiative to develop a standard for access control and authorization systems. Most of the current systems implement access control and authorization in a proprietary manner [42]. XACML provides a policy language which allows administrators to define the access control requirements for their resources in a standard and portable way. It also provides a mechanism that offers much finer granular access control than simply denying or granting access. XACML architecture is tightly intertwined with SAML architecture. While SAML addresses authentication and provides a mechanism for transferring authentication and authorization decisions between cooperating entities, XACML focuses on the mechanism for arriving at those authorization decisions [42]. The importance of XACML is that it can be used to express policies for different components of the same organization. Being able to express policies of different components in the same way is a big advantage for any organization [41].

The main actors in the XACML are PEP, PDP, PIP and PAP. Policy Enforcement Point (PEP) is system entity that performs access control by making decision requests and enforcing authorization decisions. Policy Decision Point (PDP) is system entity that evaluates applicable policy and renders an authorization decision. Policy Information Point (PIP) is system entity that acts as a source of attribute values. Policy Administration Point (PAP) is system entity that creates a policy or policy set.

The main components of policy language model are: Rule, Policy and Policy Set. A Rule is a defined as "A target, an effect and a set of conditions". A target is defined as the space of decision requests that refer to actions on resources by subjects. An effect is either permit or deny. The conditions involve the calculation of attributes of the subject, the resource, the action or the environment. Conditions make the rule dynamic. A Policy consists of a target, a rule combining algorithm, a set of rules and obligations. A rule combining algorithm specifies the procedure by which the results of evaluating the component rules are combined when evaluating the policy. A Policy Set consists of a target, a policy combining algorithm, a set of policies and obligations. A policy combining algorithm specifies the procedure by which the results of evaluating the component policies are combined when evaluating the policy set. The obligations are operations specified in a policy or policy set that should be performed by the PEP in conjunction with the enforcement of an authorization decision [41]. The usage of XACML policy language model in the framework has been explained in Chapter 4.

1.7 Organization of the thesis

The thesis is divided into seven chapters. A brief outline of each chapter is given below.

First chapter introduces the concept of grid computing, grid and web services, similarity and differences between them, their architecture, the standards and specifications available and used in designing middleware for them. The chapter discusses the role of XML, SOAP, WSDL and UDDI in grid systems. It also presents web services security related specifications like WS-Security, WS-Trust, WS-SecureConversation, XACML, and SAML *etc.*

Second chapter is 'Literature Review'. It presents details on background and related work in the areas of authentication, privacy, trust and authorization in grid systems. A host of grid security requirements like single sign-on, delegation, credential management, privacy, policy, trust *etc.* have been discussed. Work done in the areas of grid security fields like authentication, privacy, trust and authorization is presented along with the middleware available for them. Important grid services security architectures available and the features provided by them have also been presented. This chapter also discusses the limitations of existing approaches and then introduces the objectives of the thesis.

Third chapter identifies and defines elements/entities present in a grid environment. The elements like subject, service, resource, domain, filter, privacy index, purpose, privacy controller, trusted third party *etc.* have been defined and explained in this chapter. The notation used to represent these elements is also discussed. The security policy framework is based on these elements. In a typical grid environment, these elements interact with each other in a complex manner.

Fourth chapter categorizes and explains different types of security policies that need to be addressed by the security framework. The security policies have been categorized into access control and non-access control policies. The focus of the thesis is on describing mechanisms to express, evaluate and enforce access control policies as they play key role in determining access to grid services/resources. Access control policies have further been categorized into authentication policies, privacy policies, trust policies and authorization policies. The chapter describes methods for expressing and exposing these policies in the environment. The additions that have been made to represent unique features of privacy and authorization policies have also been discussed.

Fifth chapter describes the integrated security policy framework. The chapter also presents and explains the individual authentication, privacy, trust and integrated policy based authorization model along with implementation details. The authentication model provides support for single sign-on and delegation features using proxy certificates and a credential management service to store, retrieve and update multiple user credentials. The privacy model provides support for anonymous access, hidden service access and access to private information based on conformance to privacy policies. The trust model provides support for calculating direct as well as recommended trust to determine trustworthiness of target service/resource. The integrated policy based authorization model provides access to grid services based on conformance to various types of security policies. Important security features and services provided by these models have also been presented.

Sixth chapter presents the results of the different implementations. It presents important authentication, privacy, trust and authorization related scenarios that have been implemented. The chapter also discusses the performance analysis of authentication, privacy, trust and authorization policies. Explanation involving analysis and comparison of results is also presented in this chapter.

Finally, chapter seven concludes with the summary of the research carried out, the benefits and limitations of the implemented framework and future scope.

Chapter 2

Literature Review

This chapter starts with the description of grid security requirements and the framework available/used to address these requirements. Then it presents background and related work in the areas of authentication, privacy, trust and authorization in grid systems along with the middleware available for them. The chapter also discusses the limitations of existing systems and then introduces the objectives of the thesis.

2.1 Grid Security: The Multidimensional Problem

Grid is a large scale resource sharing environment where users from different administrative domains provide and use resources among each other according to established policies. As users belong to different administrative domains and there is no centralized control, security becomes a big and challenging issue in this type of environment. Grid security is a multidimensional problem. It deals with issues like authentication, confidentiality, integrity, authorization, privacy, trust, policy *etc.* The nature, scope and applicability of these issues differ in grid systems compared to traditional client server based distributed systems. Grid systems present unique security requirements that require new technical approaches. This chapter presents a comprehensive study of grid security requirements and the approaches/mechanisms available/used to address them along with their shortcomings/limitations. Section 2.1.1 discusses grid security requirements and Section 2.1.2 discusses the security frameworks available to address these requirements. We have categorized different security requirements of grid systems under four categories namely authentication, privacy, trust and authorization. Sections 2.2 to 2.5 discuss the work done in these areas. Section 2.6 formulates the problem and Section 2.7 presents the objectives of the thesis.

2.1.1 Grid Security Requirements

The first most notable effort to identify and define unique security requirements for grid systems was made by Ian Foster *et al.* [24]. According to them, in addition to satisfy standard security requirements like authentication, access control, integrity, privacy and nonrepudiation, the security architecture must also provide support for single sign-on, delegation, protection of credentials, interoperability with local security solutions, exportability, uniform credentials/certification infrastructure, secure group communication and multiple implementations. They defined single sign-on as a feature which enables users to authenticate once and initiate computations that require resources on other sites, without further authentication. Delegation is the ability of a user to delegate some/all of his/her access rights to another entity so that the delegate can act on the original user's behalf. Interoperability with local security solutions enables local security policies to remain unchanged to accommodate inter-domain access. Exportability issue means that the security policy should not directly or indirectly require the use of bulk encryption. They also require security architecture to use a uniform credentials / certification infrastructure along with the mechanisms for protecting user credentials. The support for secure communication between dynamic groups and support for multiple implementations were also identified as important grid security requirements. The security policy must not be dictating a specific implementation technology rather it should be possible to implement the security policy with a range of security technologies, based on both shared and public key cryptography [24].

Another significant effort to describe security requirements for open grid services was started by GGF OGSA Security Workgroup [43]. According to OGSA Security Workgroup, the security challenges encountered in a grid environment can be grouped into three categories: i) Integration ii) Interoperability and iii) Trust Relationships. Following paragraphs discuss each of these challenges in brief.

Integration Challenge

Integration challenge requires security architecture to be implementation agnostic. There are numerous security frameworks, standards and implementations available today. The majority of organizations and individuals have their own preferences about the security requirements that are most suitable for their own environment. This places burden on the

participants to honor the existing security frameworks and/or seamlessly integrate with them. So security architecture should be "implementation agnostic" so that it can be instantiated in terms of existing security mechanisms. Security architecture should be "extensible" also so that it can incorporate new security services when available and capable of integration with the existing security services [43].

Interoperability Challenge

Interoperability challenge deals with platform independence issues at various levels. Services that traverse multiple domains and hosting environments should be able to interact with each other, thus introducing the need for interoperability. At protocol level, different domains exchange messages across their protocol layers so they need to have interoperability at each layer of the protocol stack. Mechanisms should be there to allow domains to exchange messages in a platform neutral way. At policy level, secure interoperability requires that each party should be able to specify any policy it may wish in order to engage in a secure conversation. The policies expressed by different parties should be mutually comprehensible. At identity level, mechanisms are required to identify a user from one domain in another domain. For any cross domain invocation to succeed in a secure environment, the mapping of identities and credentials to the target domain identity is also required [43].

Trust Relationship Challenge

Trust relationship challenge deals with trust related issues among grid participants. Grid service requests can span multiple security domains. Trust relationships among these domains play an important role in the outcome of such requests. A service needs to make its access requirements available to interested clients so that they understand how to securely request access to it. The trust among the participants in a dynamic virtual organization is a complex thing to achieve and this trust must be evaluated for each session or request. This requires federation of trust relationships among the participants. The dynamic nature of the grid in some cases can make it impossible to establish trust relationships among sites prior to application execution. Given that the participating domains may have different security infrastructures, it is necessary to realize the required trust relationships through some form of federation among the security mechanisms [43].

Other efforts that also present unique grid security requirements are described in [10], [23], [44], [45], [46], [47], and [48]. According to the literature available in these papers, the security requirements for grid systems can be categorized into the following security disciplines:

i) Authentication

ii) Delegation

iii) Single Sign On

iv) Credential Lifespan and Renewal

v) Confidentiality

vi) Message Integrity

vii) Non repudiation

viii) Secure Logging

ix) Privacy

x) Trust

xi) Policy Exchange

xii) Authorization

xiii) Assurance

xiv) Manageability

Following paragraphs briefly discuss each of these security requirements:

Authentication: Authentication deals with verifying the identity of the requester. It ensures that the requester actually is the identity which he/she is claiming to be. As multiple authentication mechanisms exist, grid environment must provide integration points for them and the means for conveying the specific mechanisms utilized in any given authentication operation. The authentication mechanism may be a custom security mechanism or an industry standard technology. The authentication plug point must also be agnostic to any specific authentication technology [6], [10], [43].

Delegation: It deals with delegation of authority from one entity to another. Grid environment must provide facilities for delegation of access rights from requesters to the services. Care should be taken while delegating authority to ensure that the authority transferred through delegation is scoped only to the task(s) intended to be performed and within a limited lifetime to minimize the misuse of delegated authority [6], [10], [43].

Single Sign On: Single sign-on means enabling a user to sign on once, and then, without having to sign on again, access different domains that would normally be outside the scope of the primary sign-on domain. This capability allows a service user to utilize

multiple resources with one explicit logon process, and thereafter, automatically delegate the same authenticated credential for the next resource access without user intervention, within a specific period of time. The single sign-on sessions may also include access of resources in other domains using credential delegation [6], [10], [43].

Credential Lifespan and Renewal: In grids, credentials have limited time span associated with them and most of the grid jobs take more time to execute. This may cause credentials to get invalidated and bringing the system to an invalid state. In order to avoid this, a grid system must support credential expiry notifications to users and credential revalidation facilities [6], [10], [43].

Confidentiality: Confidentiality deals with the confidentiality of the messages or information that flow over the transport mechanisms. The confidentiality requirement includes point-to-point transport as well as store-and-forward mechanisms [6], [10], [43].

Message Integrity: It provides mechanisms to detect unauthorized changes to messages. The use of message or document level integrity checking is determined by the policy, which is tied to the offered quality of service [6], [10], [43].

Nonrepudiation: Nonrepudiation guarantees that a user can't deny at a later stage that he has not performed a particular action in case he has actually performed that particular action [6], [10], [43].

Secure Logging: Secure logging is the foundation for addressing requirements for notarization, non-repudiation, and auditing. It deals with providing all services, including security services themselves, with facilities for time-stamping and securely logging any kind of operational information or event in the course of time [6], [10], [43].

Privacy: Privacy security requirement deals with protection of private information/data. It brings privacy semantics to a service usage session. Security architecture must allow both, the service requester and the service provider to specify and enforce privacy requirements and policies [6], [10], [43].

Trust: Trust security requirement deals with trust related aspects like specifying trusted credentials, trust policies and determining trustworthiness of target service. Security architecture must allow both, the service requester and the service provider to specify and enforce trust requirements and policies [6], [10], [43].

Policy Exchange: Policy exchanges allow clients and services to dynamically exchange policy information to establish a negotiated security context between them. Such policy information may contain authentication requirements, supported functionality, constraints, privacy rules *etc.* [6], [10], [43].

Authorization: Authorization is concerned with what a requester is permitted to do. It deals with issues like who can access what services and under what conditions. Authorization model should allow access to services based on established authorization policies. Authorization model should also accommodate various access control frameworks and implementations [6], [10], [43].

Assurance: Assurance provides means to qualify the security assurance level that can be expected of a hosting environment. This can be used to express the protection characteristics of the environment such as virus protection, firewall usage for Internet access, internal VPN usage, *etc.* [6], [10], [43].

Manageability: manageability provides manageability of security functions such as identity management, policy management, security key management and other critical aspects [6], [10], [43].

In the security policy framework, the key security requirements have been categorized into four major categories related to authentication, privacy, trust and authorization. Under the authentication category, the security requirements like single sign-on, delegation, credential management, confidentiality, integrity, non-repudiation and secure communication have been included. Privacy and trust security requirements deal with privacy and trust related aspects between service providers and requesters. Authorization deals with authorization, policy specification and access control mechanisms. Assurance and manageability security requirements are not included under these categories and can be considered separately.

A number of architectures have been proposed that addresses some/all of these security requirements. Important among these architectures, along with their advantages and disadvantages, are explained in the next section.

2.1.2 Grid Security Frameworks

The security requirements described in the previous section require new technical approaches for their implementation in grid systems. The most significant security architecture for grid systems was proposed by Ian Foster *et al.* [24]. This architecture defines protocols to handle several unique grid security requirements and is the basis of GSI (Grid Security Infrastructure) in earlier versions of Globus Toolkit®. The architecture is based on security policy (a set of rules) that define the security subjects

(*e.g.* users), security objects (*i.e.* resources) and relationship among them. The security policy provides a context for security architecture within which the architecture defines protocols that govern the interaction between subjects and objects. Subjects are users and processes. Objects include the wide range of resources like computers, data repositories, networks, display devices and other peripherals *etc.* [24]. Figure 2.1 shows the overview of this architecture.

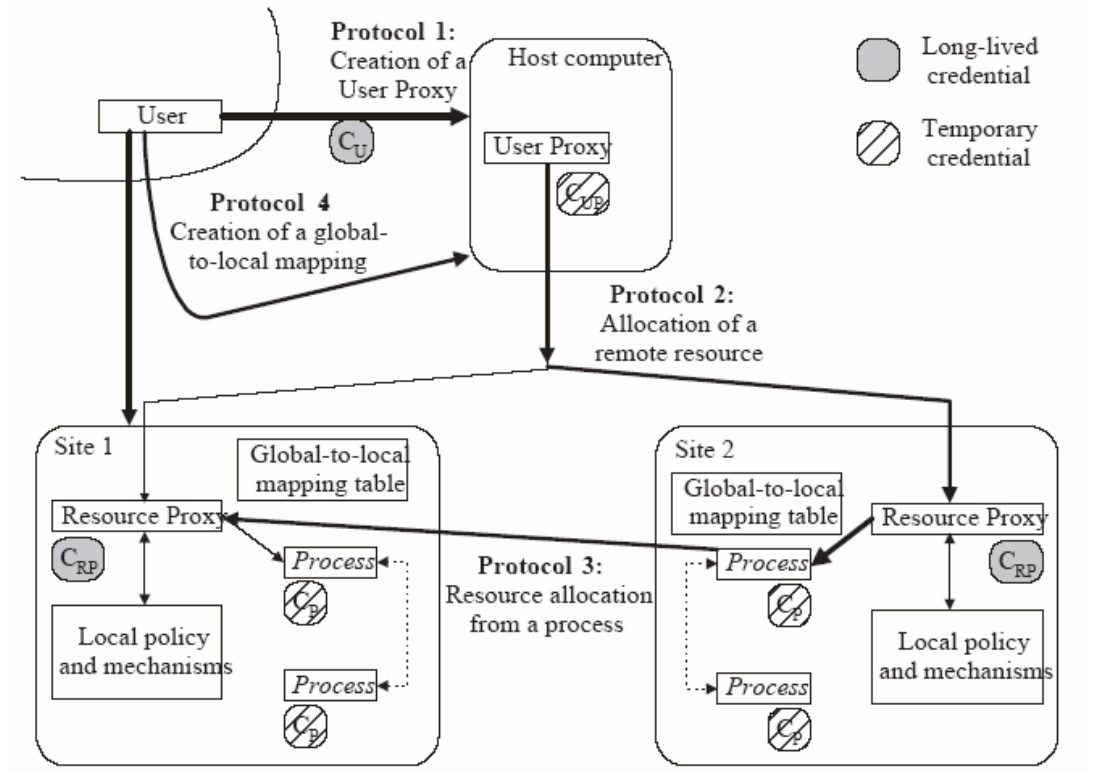

Figure 2.1: A Computational Grid Security Architecture [24]

The major components of the architecture are entities, credentials and protocols. The thick lines represent the protocols. The dashed lines represent the authenticated interprocess communication. The curved lines separating the user from the rest of the environment signifies that user may disconnect once the user proxy has been created.

First protocol of this architecture is user proxy creation protocol which introduces the concept of user proxy. User proxy can act on a user's behalf without requiring user intervention. It acts as a stand-in for the user. It has its own credentials, eliminating the need to have the user on-line during a computation and eliminating the need to have the user's credentials available for every security operation.

Second protocol is resource allocation protocol which introduces the concept of resource proxy. Resource proxy acts as an agent to translate between inter-domain security operations and local intra-domain mechanisms [24]. User proxy allocates resources using second protocol.

Third protocol is resource allocation from a process. While resource allocation from a user proxy is necessary to start a computation, the more common case is that resource allocation will be initiated dynamically from a process created via a previous resource allocation request. Third protocol defines the process by which this can be accomplished. Using this protocol a process created can allocate additional resources directly.

Finally the fourth protocol which is mapping registration protocol is used to define a mapping from a global to a local subject. The conversion from a global name into a local name is achieved by accessing a mapping table maintained by the resource proxy.

At time when security architecture defined in [24] was proposed, there were considerable differences between grid and web based systems. The original architecture as defined in [24] is not consistent with web services security specifications. Today, the grid services standards are converging with web services standards. A significant overlap exists between grid and web services security requirements. So the architecture proposed in [24] can be examined further for its applicability in today's grid and web services based world. The architecture also proposes to develop techniques for more flexible policy based access control mechanisms. We believe that the technologies and specifications developed for web services security can be used to realize this architecture in today's grid and web services based world. The Globus Toolkit® version 2 (GT2) is based on original architecture proposed in [24] but after that the newer versions, GT3 and GT4, of Globus Toolkit® are based on web services specifications [49]. Following paragraph briefly discusses web services security specifications, used for addressing important security related concerns of web services.

Web services security specifications tries to present a broad set of specifications that cover different security requirements like authentication, authorization, privacy, trust, integrity, confidentiality, secure communication channels, federation, policy, delegation, and auditing across a wide spectrum of application and business topologies [33]. WS-Security is the cornerstone of this effort and is submitted by IBM and Microsoft to OASIS (Organization for the Advancement of Structured Information Standards) [23]. WS-Security provides a SOAP messaging framework to integrate and support various existing security models and it proposes a set of extensions to SOAP to implement

integrity and confidentiality. Although WS-Security does not in itself provide a complete security solution, it defines building blocks that web services' developers can use to build security frameworks [23]. Figure 2.2 shows web services security specifications.

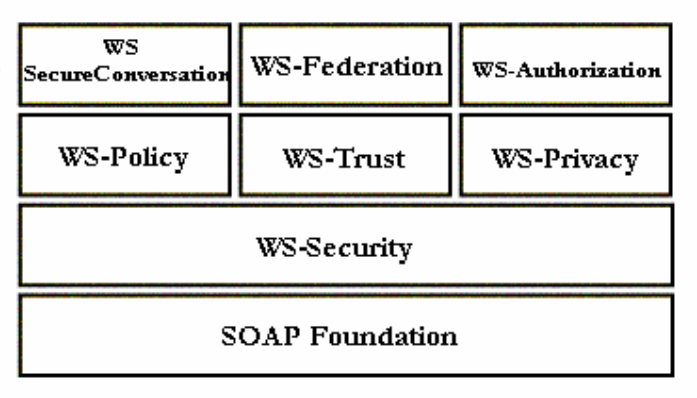

Figure 2.2: Web Services Security Specifications

This set includes a message security model (WS-Security) that provides the basis for other security specifications. These specifications address several security related issues that are also applicable to grid services. So these specifications can be used to realize the grid security model described in [24]. The specifications addresses security issues like how to associate security tokens with messages (WS-Security), how to express constraints and requirements of a web service (WS-Policy), how to request and issue security tokens to establish trust (WS-Trust and WS-Federation) *etc.* but does not give a comprehensive security architecture to address security problems present in a typical grid and web services environment. Most of other available security architectures for grid are either problem specific or not based on web services architecture. The advantage of using web services security specifications based architecture is that the tools and technologies developed for web systems can be used for grid systems also. The implemented security policy framework is making use of web services security specifications.

Another security model for grid services is proposed by GGF OGSA Security Workgroup [43]. In addition to describing unique grid security requirements, it presents a layered architecture for addressing different security related issues within Open Grid Services Architecture (OGSA). It defines a comprehensive grid security architecture that supports, integrates and unifies popular security models, mechanisms, protocols, platforms and technologies in a way that enables a variety of systems to interoperate

securely [43]. This security architecture is intended to be consistent with the security model that is currently being defined for the web services framework. The various components of this security model are shown in Figure 2.3.

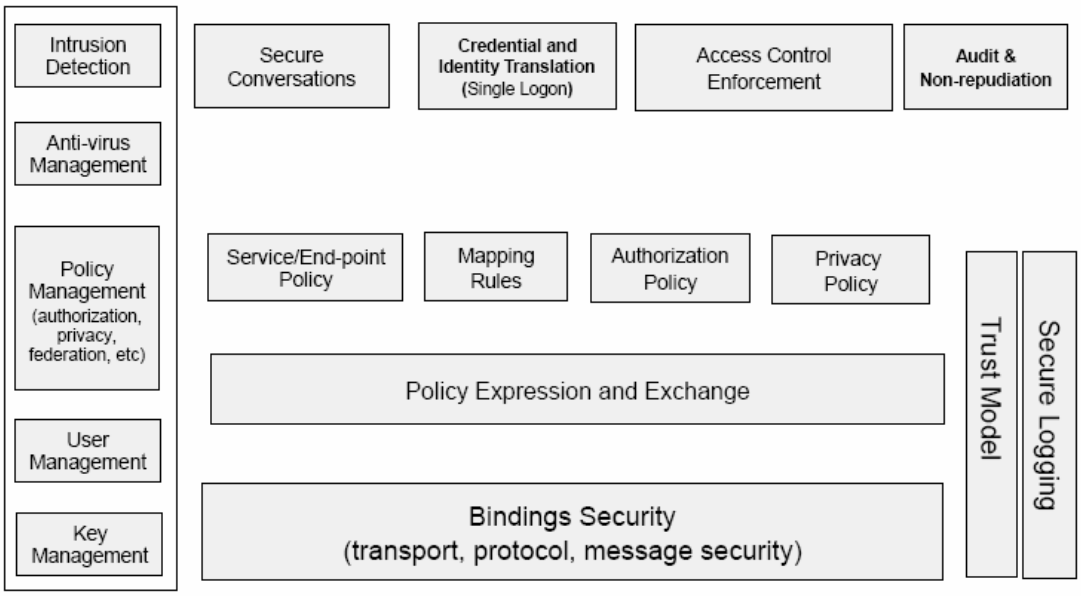

Figure 2.3: Components of Grid Security Model [43]

The base layer of this model is binding security that deals with transport mechanisms. In this model, application specific components such as secure conversation, credential identity translation, access control enforcement, and audit and non-repudiation depends on policies and rules for authorization, privacy, identity/credential mapping, and service/end point [50]. In order to apply and manage these policies, one needs a language for policy expression and exchange and means to securely communicate through bindings to transport protocols [50]. To realize the functionality of this layer, WS-Policy or XACML specifications can be used to express and exchange policies among different entities. One side of this model has trust model that define trust anchors and how trust is derived. Other side of this model has management components that are also subject to policy enforcement [50]. This framework proposes a layered architecture but does not provide specific models to handle authentication related issues, privacy requirements, trust management and authorization and access control related issues.

There are some other efforts also to define security architecture for grid systems. Important among them include Legion [51], CRISIS [52], Global Security Architecture

[53]. Others are [54], [55], [56], [57], [58] and [59]. Some of these are based on the above systems in one or other way.

Legion is a distributed computing platform that combines large collections of independently administered machines into single, coherent environment. It is composed of independent, active objects. All entities within the system are represented by objects. Every object in a Legion system is identified by a unique, location-independent Legion Object Identifier, called LOID. One of the LOID fields stores an X.509 certificate. Legion users are given LOIDs which are registered with the system and entered in appropriate system groups and access control lists by resource providers. When an object makes a call on behalf of a user, the user's LOID and associated credentials provide the basis for authentication and authorization. In Legion, access is the ability to call a method on an object. The general model for access control is that each method call received at an object passes through a layer called "MayI" before being serviced. "MayI" decides whether to grant access according to whatever policy it implements [51]. Legion provides strong authentication and authorization capabilities but does not address privacy and trust related issues among different objects.

CRISIS [52] is a wide area security system that defines a uniform and scalable security infrastructure for wide area systems but does not address interoperability with local security mechanisms. It is also not consistent with web services specifications. The Global Security Architecture [53] presents a high level view of the overall architecture planned to be implemented in EGEE project. The security architecture inherits many of the thoughts from previous projects and parallel ongoing efforts. It is an evolutionary continuation of previous projects and efforts with new components and features added due to transition to a service oriented environment but it does not provide any specific model to address privacy and trust related issues among different entities in a grid environment. Quan Zhou *et al.* [55] propose a scalable, layered security architecture for grids but it is not consistent with web services specifications. Yuri Demchenko *et al.* [56] propose Security Architecture for Open Collaborative Environment (OCE) with the intent to build a flexible, customer-driven security infrastructure for open collaborative applications. The architecture is based on extended use of emerging web services and grid security technologies combined with concepts from the generic Authentication Authorization and Accounting (AAA) [60] and Role-based Access Control (RBAC) frameworks but this architecture also does not provide specific models to handle complex privacy and trust related issues among service providers and consumers. Similarly the

models proposed by Debasish Jana *et al.* [54], Xiuying WU *et al.* [57], D. Agarwal *et al.* [58] and Syed Naqvi *et al.* [59] provide basic frameworks for handling security issues emphasizing on either one or two of the aspects from authentication, privacy, trust and authorization but none of them provide integrated support to handle authentication, privacy, trust and authorization related issues in a single framework.

A lot of other efforts are focused on defining frameworks for handling one of the issues from authentication, privacy, trust and authorization. Following sections discuss these issues and work done in these areas along with the available middleware.

2.2 Authentication

Authentication is the first and most important link in grid security chain. It deals with verifying the identity of a user. It ensures that the requester actually is the identity which he/she is claiming to be. Authentication is achieved through the presentation of some security credentials that cannot be forged [47]. In traditional client server based approaches, users are generally given some credentials that they present to the server to access the services. The server verifies the authenticity of the credentials and based on the result, it either allows or denies access to the service. Following sections briefly present the mechanisms used for authentication, problems related to authentication and the authentication solutions implemented by different middleware.

2.2.1 Authentication Schemes

The authentication problem is generally handled using three different mechanisms: based on shared secret, based on public key and based on third party. Following paragraphs discuss each of these mechanisms.

Shared secret based authentication

This mechanism works by sharing a secret. Password authentication is the most commonly used method under this scheme. A user shows its id/username and password and the server checks the id: password pair by referring to a user registry that manages a collection of such pairs. If id: password pair appears in user registry then authentication

succeeds otherwise fails. One of the advantages of this mechanism is its simplicity. It is computationally inexpensive also as the password supplied by the user is checked with the hash of the password which is stored in the registry. But the limitation is that it can't be practically used to implement single sign-on and delegation requirements. So it is generally not used in grid systems.

Another way to implement the shared secret is through a challenge mechanism. Here the authenticator challenges the user to encrypt a bit of known information using the shared key. The user responds by encrypting the required information which is then validated by the authenticator. This mechanism is slightly more expensive than password authentication. The shared secret also needs to be changed periodically so that adversary cannot guess the secret.

Public key based authentication

In this type of scheme the user has a public and private key pair and the authenticator knows the public key of the user. The user encrypts the information with his private key. The authenticator can verify the authenticity by decrypting the same information with user's public key. This type of system is very secure and generally temper proof. The biggest problem in adopting the system in wide scale is the scalability of the system. If the system has thousands of users then it is very difficult for authenticator to maintain public key information of so many users. So in reality, a variation of this scheme is used which is called the certificate based system or third party authentication system [61].

Third party authentication

In this scheme the user gets a digital certificate from an entity called Certificate Authority (CA) which is known as third party. Certificates are nothing but information about the user hashed and then signed by the CA's private key. Since the public key of the CA is widely known, the authenticator has no problem in validating the certificate and hence authenticating the user to access the system based on certificate. However, this scheme requires the authenticator to trust CA. The system also requires PKI (Public Key Infrastructure) for this scheme to work. This may not be feasible always. Another mechanism of third party based authentication used in Kerberos system is to have a Key Distribution Center (KDC) which authenticates the user using a standard mechanism like

using a password. The KDC generates a session key for the user to access the system encrypted with the systems public key.

These are the basic authentication schemes and their variations are commonly used in almost all types of distributed systems including wired or wireless systems [62]. In grid systems, PKI and Kerberos are the commonly used security infrastructures. PKI provides a basis to certify holders of public keys. The key constructs of PKI are the certificate and the certificate authority. A certificate is a proof of identity. A Certificate Authority (CA) is an entity that issues certificates. With a certificate, an entity can be related to its public key. Because the certificate is digitally signed, its contents can be trusted as long as the certificate issuer is trusted. The certificate format can be X.509, PGP (Pretty Good Privacy) or any other custom certificate format. In grid systems, X.509 certificates are mainly used. PKI uses asymmetric key operations. The main problem with asymmetric key operations is that they are slower than symmetric key operations. So if performance is a major concern, then symmetric key operations can also be used. In symmetric key operations each side uses same the keys based on which secure link is established to exchange secret information. In asymmetric key operations each side sends his public key and signs messages with his private key. Now each side can authenticate the other directly provided the public keys exchanged really belong to the parties involved. To ensure this, each side has its public key signed by a Certificate Authority. On the grid, the normal practice at present is for both sides to use certified public keys for authentication using X.509 certificate format.

2.2.2 Additional security requirements related to authentication

Other problems related to authentication (in grid systems) are of single sign-on, delegation and management of credentials. The execution of a grid job may span over many administrative domains. If a grid job is supposed to execute over multiple domains (on user's behalf) after its submission to first domain, then security credentials must be passed to other domains also. For this, the user is required to sign in multiple times (once for each domain). This situation is undesirable. Support must be there for single sign-on *i.e.* the user should login just once and then be able to access the service/resources of other domains that come in the execution chain without further intervention.

Delegation is also an important requirement in grid systems. The execution of a grid job may take hours or weeks or months. So it may not be possible for users to remain online for such long times. In these situations users must be able to delegate some/all of their rights to services, so that those services can execute job on user's behalf. In grid systems, single sign-on and delegation requirements are achieved through the use of proxy certificates [63], [64].

Next problem is of management of multiple user credentials. In grid systems, the security requirements of different services vary widely with respect to the type of security credentials they need to authenticate users. Different services require different credentials like username: password, X.509 Certificates, Kerberos Tickets, Custom Security Tokens *etc.* to verify requesters. This is because services in a virtual organization are provided by service providers that belong to different physical organizations and they are free to implement any authentication mechanism of their choice to protect their services/resources. If we allow users to use different security credentials to access different services then the main problem that comes is the management of these credentials *i.e.* how and where they are to be stored, how they are to be retrieved according to the requirements of the service, how they are to be delivered to the service *etc.* So support must be there for management of multiple user credentials.

Besides these, the support to express, evaluate and enforce authentication policies should also be there. WS-Security is the most commonly used specification to describe authentication requirements of a service. It can be used to express simple authentication policies like specifying security credentials required to access a service, encryption and signature requirements of a service *etc.* But other authentication policies that include access control related information cannot be expressed easily in WS-Security. *e.g.* a policy might state that users of domain-A must have X.509 certificates and users of domain-B must have username: password in order to access a particular service. A policy might also state that a service can be accessed through username: password only on Sundays but on all other days it require X.509 certificates. The enforcement of these types of policies requires integration with access control mechanisms. So support must be there to express, evaluate and enforce authentication policies also.

Other security requirements that can be addressed along with authentication system are of nonrepudiation, integrity and confidentiality of messages. These security requirements can be handled through encryption (symmetric as well as asymmetric

techniques) and digital signatures. A lot of work has already been done in these areas, so existing techniques can be used to address these security requirements.

2.2.3 Authentication in existing middleware

A number of important projects address the challenges related to authentication. Major among them include Microsoft .NET Passport system [65], Entrust TruePass portfolio [66], MyProxy [67], Kerberized Certificate Authority (KCA) [68], Password Safe [69], Microsoft Trust Bridge [70], Liberty Alliance Project [71], CredEx [72] *etc.*

The goal of Microsoft .NET Passport system [65] is to provide a uniform and distributed network to perform authentication. Passport implements single sign-on solution for the web by providing users one set of credentials (id, key, *etc.*) which are centrally stored. To use single sign-on, the target service must be Passport enabled. A user can carry a single Passport credential among several web applications that support the Passport scheme. In this manner, the user need only manage a single login name and password in order to access disparate applications. The user authenticates to Passport, Passport provides a cookie, and the user presents this cookie to the web application. Thus it functions much like Kerberos tickets [40]. It does not support proxy certificates. In our authentication framework, credentials are distributed (no central repository) over Domains (explained in chapter 5), and it supports the use of proxy certificates.

The Entrust TruePass portfolio [66] is another system that promises convenient web based authentication using PKI based Entrust Digital IDs and signatures. It also requires the target web services to be TruePass enabled. Entrust TruePass is an enhanced Web security solution that leverages the advanced public-key infrastructure (PKI) capabilities provided by the Entrust Authority™ Security Manager. It enables the protection of web site resources and applications and adds accountability and privacy to online transactions [66]. But in this credential management and delegation features are not supported.

MyProxy [67] on the other hand is a true credential repository and supports the storage of multiple user credentials. It is developed to meet the credential management requirements of the grid community. The MyProxy system is the implementation of the Virtual Soft Token system proposed in [73], where the X.509 proxy certificates are used to store and retrieve user credentials without having to expose the private key. MyProxy is being extended with a name change to GridLogon to reflect its enhanced capabilities.

The main function of the GridLogon service is to issue X.509 credentials to clients that are used by the client to securely access Globus Toolkit® enabled Grid resources [74].

KX.509 1.0, from the University of Michigan, is a Kerberized client-side program that acquires an X.509 certificate, using existing Kerberos tickets. The Globus Toolkit® [75] normally uses X.509 credentials [76], [77]. KX.509 allows a user to authenticate to a host that is running Globus software using Kerberos tickets instead of requiring X.509 certificates to be installed [68]. X.509/KCA is a system that just supports the exchange of Kerberos ticket-granting ticket for X.509 proxy certificate but it is not a credential repository.

Another system Password Safe [69] is an example of a password database that allows users to store passwords for different accounts. Password Safe protects passwords with the Blowfish encryption algorithm. The problem with Password Safe is that it involves the exchange of a single type of token only. It is not a single sign-on and delegation solution.

The Liberty Alliance Project [71] is an effort, initiated by Sun Microsystems and now involving many companies, with the goal of facilitating authentication on the web for both human users and automated mechanisms. Liberty Alliance Project defines how various identity providers (which store information related to user identities) can be linked together to support single sign-on across multiple distinct services. The framework provides standards for sharing attribute information across a trusted set of entities called the Circle of Trust (COT). The user is authenticated by the identity provider and after authentication, user can access services provided by service providers. The sessions are transferred between different service providers. The user may possess different identities with different service providers as long as they are connected through the circle of trust constraint. The weakness of Liberty Alliance Project is that it falls short of addressing support for diverse authentication token types [72].

Microsoft's TrustBridge [70] is federated identity effort that builds on WS-* set of web services security specifications like WS-Security, WS-Trust, WS-Federation *etc.* TrustBridge enable businesses to share user identity information between applications and organizations. It allows different organizations using the Windows operating system to exchange user identities and interoperate in heterogeneous environments using industry-standard XML web services protocols.

CredEx [72] is an open source standards based web service that facilitates the secure storage of credentials and enables the dynamic exchange of different credential types

using WS-Trust token exchange protocol. With CredEx, a user can achieve single sign-on by acquiring a single credential but support for credential delegation is not there. MyProxy [67] and CredEx [72] are indeed a key inspiration for the authentication framework.

More attempts in this area also exist like [78], [79], [80], [81] and [82] but these do not support all types of authentication features required in grid systems. The implemented authentication framework is distinguishable from the discussed authentication systems in different ways *e.g.* the implemented framework is based on web services security specifications like WS-Security, WS-SecureConversation *etc.* It supports the storage of multiple types of tokens/credentials. (*e.g.* X.509, Kerberos, Custom Security Token *etc.*). It can be used for grid as well as web services. It provides a credential manager service which is distributed over different domains so removes the drawbacks of central storage. It provides support for single sign-on and delegation through proxy certificates. It provides a mechanism to express, evaluate and enforce authentication policies also. These are some of the characteristics of the implemented authentication framework that make it different from other systems.

2.3 Privacy

Privacy is becoming a serious concern in service oriented environment like grid and web, where a large number of users provide their general as well as personal information to service providers to gain access to their services. From service requester's point of view, privacy deals with ensuring that service providers are not misusing service requester's data and are using the data only for the purpose for which it has been submitted. Service providers generally publish their privacy policies in plain English language to ensure users that information provided by them will not be used inadequately. But the claims made by them (which are written in plain English) can't give users the guarantee that their information will not be misused [83]. The privacy model implemented by service providers must have the ability whereby uses have control to check the misuse of their data as well as control over how their information can be collected, stored, used and shared.

2.3.1 Privacy Issues

A Privacy Model should address a wider notion of privacy that focuses on providing users with the purpose of data collection, choice regarding collection and conditions under which data can be used so that users can control and make decisions regarding the disclosure of their private information. It must support privacy requirements like individual control, collection limitation, purpose specification and consent *etc.* [84].

In addition to the assurance of proper privacy of personal data, users generally expect more privacy features from service providers [85]. Sometimes users want to hide their actual identity while accessing a service. *e.g.* consider an online system that provides e-voting services. Here individuals may wish to hide their identity while voting for a particular party but not the fact that they are voting/accessing a service. Sometimes users want to hide even from the service provider that they have accessed a particular service. *e.g.* a user may want to hide from service provider that he has accessed a service to purchase a video titled "xyz", because, if service provider know that a particular user is accessing a particular service with particular parameters then he can observe user's usage pattern and make interpretations (data mining). So some support should be there to handle these types of privacy requirements. Though this issue can be resolved by granting anonymous access to services but this does not give a complete solution because if service providers do not know about user preferences then personalized access to services can not be provided. This is likely to disadvantage users as they will not be able to customize services according to their interests and hence will not experience quality of service. In fact, participation in the real world requires disclosure of information to some extent [86]. To get customized access to services according to our preferences, it is necessary to supply preference details (private information) to service providers.

Most of the work in privacy is oriented towards handling the privacy requirements of service requesters. Besides handling privacy requirements of service requesters, privacy concerns of service providers should also be addressed. *e.g.* consider a secret service is provided by government to some of the officials of a department. If the officials have access to this secret service then they can leak information regarding security credentials required to access the service, address of the service, input and output from the service *etc.* Clearly, government in this case requires that officials must not know the credentials/attributes required to access the service, the actual address of the service and

the input and output requirements of the service. Most of the privacy models address privacy requirements of service requesters and not of service providers. But the scenario just described demands a privacy model which must be able to handle privacy requirements of service providers also.

The access control mechanisms implemented by authorization framework also need to be extended to include privacy based access to data and services. The access control decisions should not be determined by the user's identity alone, but by the task the individual user is currently performing and his conformance to established privacy policies. Personal data should be accessible to a user only if such an access is necessary to perform his task and if he is authorized to perform this task [87].

The privacy model should also allow service providers to express, evaluate and enforce privacy policies on their resources/services to protect them. Web services security specifications propose the use of WS-Privacy to address privacy related issues between service providers and requesters but it has not been completed as of June 2008. WS-Privacy will use a combination of WS-Policy, WS-Security and WS-Trust to communicate privacy policies. The privacy policies are stated by organizations that are deploying web services and require that incoming SOAP requests contain claims that the sender conforms to these privacy policies. WS-Security specification can be used to encapsulate these claims into security tokens, which can be verified. To express access control privacy policies, XACML can also be used. XACML, as already discussed, is eXtensible Access Control Markup Language that is used to express general access control related policies. As it is extensible, it can be used to express a variety of other security policies also. In the implemented privacy framework, we are making extended use of XACML to express privacy related access control policies.

2.3.2 Privacy handling in existing middleware

Most of the software available to construct grid and web services do not provide effective methods to handle complex privacy requirements of both the service providers and the service requesters. Much of the work done in the area of privacy in distributed systems like grid is limited. There are some projects that address privacy requirements of grid systems in one or other way *e.g.* Shibboleth [88], PRIMA [89], PERMIS [90], GT4 [76]

but they are mainly policy based authorization frameworks. The efforts like P3P [91] and EPAL [92] are primarily focused on addressing privacy related issues in web systems.

Shibboleth [88] is an attempt to address privacy in an authorization environment but it is primarily focused on using pseudonymity. It does not provide a complete privacy based access control environment. It is a tool for identity federation between campuses that allows resources to obtain attributes about the user (*e.g.* departmental affiliation, student status), while preserving the user's privacy and not having to become involved with the details of how the user is authenticated in their home domain [74].

The efforts like PRIMA [89] and PERMIS [90] mainly provide policy based authorization framework but provide little support to address complex privacy related issues. They do not provide purpose based access to grid services. The most commonly used toolkit to construct grid services GT4 [76] also does not provide any comprehensive privacy model. The support is also missing from Alchemi [93] which is used to construct grid applications in .NET environment.

Another effort, P3P [91], is a XML document that describes the data collection practices for a site. It is a specification by the World Wide Web Consortium (W3C) and provides a framework for informed online interactions on the web, allowing users to negotiate agreements with services that declare their privacy practices and request data. P3P provides a vocabulary and standard machine readable format to allow websites to declare their privacy practices and describe the data they collect. The user privacy preferences are described using a P3P Preference Exchange Language (APPEL 1.0 [94]). Users can use this language to express their preferences as a set of rules (called a ruleset) which can then be used by their agent to make automated or semi-automated decisions regarding the acceptability of machine-readable policies from P3P compliant web sites [84]. It is mainly used in web based systems to publish privacy policy of an organization to requesters but it does not provide any technical mechanism to check non conformance of these policies at service provider's end. P3P specification complements the implemented privacy model in the sense that it can be used for declaring and negotiating privacy policies.

EPAL [92] is an XML-based markup language that formalizes enterprise-internal privacy policies. It approaches the problem on the server side and addresses the need of a company to specify access control policies, with reference to attributes/properties of the requestor, to protect private information of its users. EPAL is designed to enable organizations to translate their privacy policies into IT control statements and to enforce

policies that may be declared and communicated in P3P [84]. EPAL does not provide any comprehensive privacy model to support anonymous access, hidden services access and purpose based access. Moreover it does provide any procedure to integrate privacy based access control mechanisms with authorization framework.

Marc Langheinrich [95] suggests a P3P-style architecture to provide notice of data collection in ubiquitous computing environments. In this, sensors and other recording devices use a P3P declaration format to announce their data collection practices. User agents on user mobile devices release contextual information based on a comparison between the privacy declaration and user preferences encoded in a machine readable format. The proposed Privacy Awareness System (paws) implements P3P in ubiquitous computing environments to provide notice-choice based data collection.

Simone Fischer-Hubner *et al.* [87] augmented a task-based access control model with the notion of purpose and consent. Here data can be accessed in a controlled manner only by executing a task. A requester can access personal data if this access is necessary to perform its current task and he is authorized to execute this task. Additionally, the task's purpose must correspond to the purposes for which the personal data was obtained or there has to be consent by the data subjects. A task consists of a set of certified operations representing the set of "necessary accesses" to perform that task. However, the model does not consider context-dependent access control or obligations. It is also not based on web services security specifications or standards.

P. Bonatti *et al.* [96] developed a language for use-based restrictions that allows one to state under which conditions specific data can be accessed. In this language, a data user is characterized as the triple user, project, and purpose. Projects are named activities registered at the server, for which different users can be subscribed, and which may have one or more purposes. Each user and project is associated with a profile, which captures properties such as name and address or title and sponsor. However, the language does not support obligations and consent. Gunter Karjoth *et al.* [83] propose a privacy model for enterprises. The model is not based on open standards but is indeed a key inspiration for our privacy model.

Other approaches used to handle privacy requirements and policies are described in [84], [97], [98], [99], [100] and [101]. Ajay Brar *et al.* [84] describes at a higher level the framework for providing privacy-aware personalization in ubiquitous computing environments but do not present any comprehensive privacy model and its integration with authorization framework. Scott Lederer *et al.* [97] present a conceptual model for

everyday privacy in ubiquitous computing environments. Stephen Crane [98] uses the concept of trust agents for preserving privacy. Both, [97] and [98] are not based on standards and do not support purpose based access. U.M. Mbanaso *et al.* [99] propose a Trust Authorization Framework (TAF) that builds on the capabilities of XACML to support the bilateral exchange of policies and credentials through trust negotiation. The framework has the capabilities to protect resources, policies and credentials simultaneously in distributed environment for users with or without pre-existing trust relationships but does not support purpose based access. Wei Xu *et al.* [100] addresses consumer privacy concerns in the context of highly customizable composite web services. The approach involves service producers exchanging their terms-of-use with consumers in the form of "models". The framework also provides automated techniques for checking these "models" at the consumer site for compliance of consumer privacy policies but the framework does not make use any standard (*e.g.* XACML) for specifying privacy policies. It uses its own mechanism for specifying privacy policies which is based on the concept of "labels". Christian Schlager *et al.* [101] introduced the idea to use attribute based access control mechanisms with XACML policies for building a flexible and privacy aware system. They have tailored the model for b2c eCommerce applications but the model does not support purpose based access.

To our knowledge, no software is available to handle complex privacy requirements between service providers and service requesters while developing grid and web services. Many of them are not based on open standards and use their own mechanisms for handling privacy requirements and to express, evaluate and enforce privacy policies. In the implemented privacy model, we have made an attempt to define a general, standards based privacy model that is able to address the privacy issues of both, the service requesters and service providers. The model supports anonymous access and hidden services access through the concept of anonymous and custom security tokens. The model also provides mechanisms to express, evaluate and enforce privacy policies. It also describes the integration of privacy based access with authorization framework.

2.4 Trust

The context of grid computing introduces challenging trust related issues as both, resource providers and resource consumers can come from mutually distrusted

administrative domains and any of them can behave maliciously. The usage of grid system can become severely limited if participants are not given any means to access the trustworthiness of each other in the environment. To achieve faithful domain to domain interaction and confidence of participants in accessing/providing resources and services from/to other administrative domains, it is extremely important for domains to address trust issues by introducing a trust model and integrating that model into authorization framework to enable trust based access to resources and services.

Trust is a fascinating subject and social scientists have researched into the concept and developed theories around it. It can be defined in different ways at different levels. It is a subjective and elusive notion. The concept of trust is excessively complex and appears to have many different meanings depending on how it is used [102]. Trust is a multifaceted issue that may be related to other attributes such as reliability, accuracy, honesty, risk, competence, security, belief, perception, utility, benefit, expertise *etc.* [102], [103]. Trust is generally defined as having confidence that a service/resource will behave in an expected manner despite the lack of ability to monitor or control the environment in which service/resource operate [47]. Diego Gambetta [104] has defined trust as *"a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action"*. Grandison and Sloman [105] define trust as *"the firm belief in the competence of an entity to act dependably, securely, and reliably within a specified context"*. Another good definition of trust is given in [106] as: *"Trust is the firm belief in the competence of an entity to act as expected such that this firm belief is not a fixed value associated with the entity but rather it is subject to the entity's behavior and applies only within a specific context at a given time"*. This definition captures many important properties of trust as: i) Trust is not a fixed value, it is dynamic. ii) It is context and time dependent. iii) It depends on entity's past and present behavior. Trust can also be affected by those actions that we cannot monitor. Another property of trust is that it is not transitive. *i.e.* if A trusts B and B trusts C then it does not mean that A trusts C. Transitivity may hold if certain conditions are met but in general, trust is not always transitive [107].

2.4.1 Taxonomy of Trust

This section describes important terms and concepts related to trust like trustworthiness, reputation, feedback, opinion, trust functions, their categories, types of trust *etc.* as defined in [48] and [108].

Trustworthiness

An entity's trustworthiness is an indicator of the quality of the entity's services. It is often used to predict the future behavior of the entity. If an entity is trustworthy, it is likely that the entity will provide good services in future transactions also. Trustworthiness is generally given a value from 0 to 1. However, trustworthiness can be a continuous function also.

Reputation

Reputation indicates the general perception of a system's trustworthiness as perceived by other entities. The concepts of reputation and trust are closely related. Reputation metric is generally used by individuals based on word-of-mouth, or past history to determine the trustworthiness of a person or other things. Alfarez Abdul-Rehman *et al.* [107] define reputation as *"an expectation about an individual's behavior based on information about or observations of its past behavior."* In online communities, where an individual may have much less information to determine the trustworthiness of others, their reputation information is typically used to determine the extent to which they can be trusted. Similarly, in a distributed grid environment, where the grid nodes join or leave the grid, the reputation information can be used to determine the trustworthiness of the nodes.

Feedback

Feedback is a statement given by the user about the quality of a service provided by a service provider. Feedback can be multidimensional, reflecting the user's evaluation on a variety of aspects of a service, *e.g.*, price, product quality *etc.* But for simplicity, it is generally taken to be one-dimensional.

Opinion

An opinion is a user's general impression about a service provider. It is derived from the feedback on all the transactions that are conducted with the service provider. Similar to feedback, opinion also can be multidimensional in nature.

Trust Function

A trust function is a set of metrics used to infer the trustworthiness of an entity. Following are the different categories of trust functions as defined in [108].

Subjective vs. Objective

An entity's trustworthiness is often related to the quality of services it provides to others. If the quality of a service can be objectively measured, then an entity's trustworthiness for that service is called objective trust. For some other services, their quality cannot be objectively measured. Sometimes, it largely depends on each individual's taste and other subjective factors. In this situation, it is only meaningful to discuss the trustworthiness of the entity from the specific source's point of view. This type of trust is classified as subjective trust [108].

Transaction based vs. Opinion based

If trust function relies on the information of individual transactions to infer an entity's trustworthiness, then trust function is called transaction based. But if it requests opinion information from others to infer trustworthiness, then it is called opinion based.

Complete vs. Localized information

Trust functions can also be classified according to the way information is collected. If trust function assumes that every entity has same access to all the transaction or opinion information, then trust function is classified as global trust function but if each entity has access to different information, based on its location or position in the environment, then trust function is categorized as localized trust function. A localized trust function is thus also subjective [108].

Rank based vs. Threshold based

If the trustworthiness returned by a trust function can be interpreted as an approximation of some of the property, on which threshold value can be defined, then trust function is called threshold based. But sometimes trustworthiness of a single entity alone does not convey much information. It becomes meaningful only when it is compared with the trustworthiness of other entities. Such trust functions return the relative ranking of an entity. These types of trust functions are classified as rank-based functions.

Types of Trust

As defined in [106], trust is classified into two categories: Identity trust and Behavior trust. Identity trust is concerned with verifying the authenticity of an entity to determine its trust level with respect to what the entity is authorized to do and what can be expected from it. Behavior trust, on the other hand, is more real and deals with a wider notion of entity's trustworthiness.

2.4.2 Approaches used for managing trust

Trust management is the activity of collecting, encoding, analyzing and presenting evidence relating to competence, honesty, security or dependability with the purpose of making assessments and decisions regarding trust relationships [102]. A trust management system for large scale distributed systems like grid and web should be scalable, reliable and secure. Two main approaches, currently available for managing trust are: i) Policy based trust management ii) Reputation based trust management. Following paragraphs briefly discuss each of these.

Policy Based Trust Management

Policy based trust management deals with using policy languages and engines for specifying and reasoning trust. The goal is to determine whether or not an unknown user can be trusted based on a set of credentials and a set of policies. The different entities or components constituting the system exchange and manage credentials to establish trust relationships which are further refined based on certain predefined policies. Policy based

trust is involved in making access control decisions. It is intended for systems where behavior is guided by complex rules and policies.

Reputation Based Trust Management

In reputation based trust, the focus is on trust computation models capable of estimating the degree of trust that can be invested in a certain party based on the history of its past behaviors [102]. It is intended for distributed systems where a system only has a limited view of the information in the whole environment. New trust relationships and trust values are inferred based on the available information. The available information is based on the recommendations and experience of other users.

Another approach to manage trust is by using Trusted Authority (TA) where the basic concept is same as that of a CA (Certificate Authority) but are specifically meant for dealing with trust. But in large scale distributed computing system like grid, its credibility and usage decreases and uncertainty increases as the community of its trustees grows [109].

A trust management system generally includes negotiation of trust when a member joins, storage of trust metrics and distribution of trust metrics. The trust life cycle is composed of mainly three different phases: trust creation phase, trust negotiation phase and trust management phase. The trust creation phase generally is done before any trusted group is formed and it includes mechanisms to develop trust functions and trust policies. Trust negotiation is activated when a new un-trusted system joins the current distributed system or group. Trust management phase is responsible for recalculating, updating and storing trust values based on transaction information.

Trust model must provide mechanisms to express, evaluate and enforce trust related access control policies. These policies involve determining the trustworthiness of the target service/service provide/domain to reach at an authorization decision. Like privacy policies, most of the mechanisms used to express, evaluate and enforce trust policies are problem specific or not based on standards. WS-Trust describes a framework for trust models that enables web services to securely interoperate but it does not tell how to express, evaluate and enforce trust based access control policies. As already mentioned, XACML can be extended to express any general access control policy, it can be used to express trust related access control policies also.

2.4.3 Trust management in existing middleware

Much of the work to enable trust relationships in grid is through the use of X.509 based digital certificates which verify the identity of the requester. In this approach trust is assumed to be associated with the identity of the requester but this approach cannot be used to address complex trust related security issues. To address a wider notion of trust, a deeper understanding of interactions among service providers and service requesters is needed which cannot be achieved by identity based trust alone. So behavior trust should be used to address a wider notion of trust.

A number of important projects address the challenges related to trust management. Major among them include KeyNote [110], PolicyMaker [111], [112], PeerTrust [113], XenoTrust [114], NICE framework [115], Secure Grid Outsourcing (SeGO) [116], [117], TrustBuilder [118], [119], [120], [107], [109], [121] and [122].

Trust models such as KeyNote [110] and PolicyMaker [111] are concerned with identity trust. The trust model proposed by Mary R. Thompson *et al.* [112] describes trust issues involving reliance on Certificate Authorities and X.509 Identity Certificates and is also identity based. These trust mechanisms do not consider behavior trust which is dynamic and changes with time and thus these approaches have no mechanism to dynamically establish and monitor complex trust relationships. PeerTrust, XenoTrust, NICE trust management system and Secure Grid Outsourcing are reputation based trust management systems.

PeerTrust [113] was developed at Georgia Tech with Peer-to-Peer based electronic applications in mind. The PeerTrust metrics are derived from a combination of parameters like feedback, total number of transactions, factor of credibility, transaction context and community context. PeerTrust does not use a centralized database for storing the trust information. Rather, the trust information is stored in a distributed manner over the network. Each peer or a node in the network has i) a trust manager that is responsible for feedback submission and trust evaluation, ii) a small database that stores a portion of the global trust data, and iii) a data locator for placement and location of trust data over the network. The trust computation is based on the computation of trust data collected from different nodes or peers.

XenoTrust [114] is another trust and reputation management architecture. It consists of three main components: XenoServer, XenoCorp, and XenoServer Information Services

(XIS). XenoServers provide services to the client. XenoCorp provides authentication, auditing, charging, and payment services. XIS is used for storing the XenoServer status updates. The XenoTrust architecture introduces two levels approach of managing trust, called authoritative and reputation based. Authoritative trust is a boolean property established between a XenoCorp and the clients and servers that register with it. Reputation-based trust, on the other hand, is a discrete continuous property which quantifies, in a particular setting, the trustworthiness that one component ascribes to another [48]. It is distributed and highly subjective, in the sense that each entity has its own, independent view of others' reputations.

NICE [115] is a platform for implementing cooperative applications over the Internet. NICE uses a distributed trust evaluation mechanism. Whenever there is a transaction between two different nodes or peers, the peer receiving the service signs a cookie showing that the peer generating that service has successfully completed the transaction. Therefore, each node maintains a trust relationship and trust value based on the transaction it has received. The node generating transaction can store the signed cookies to prove its trustworthiness sometime later.

The Secure Grid Outsourcing (SeGO) system is developed for securely scheduling a large number of autonomous and indivisible jobs to grid sites. SeGO introduces fuzzy logic based trust integration model [116], [117]. SeGO scheme does not explicitly deal with the storage of trust information.

TrustBuilder [118] presents a policy language and infrastructure for trust negotiation for open systems. In TrustBuilder protocol and architecture, the negotiating parties establish trust between themselves by negotiating trust in a need-to-know manner. In this way, all the credentials are not disclosed to the either party. As it is a policy based trust management system it does not make use of any trust function to calculate trustworthiness of different parties. [119] proposes a general-purpose, application-independent Dynamic Distributed Trust Model (DDTM). In this, access rights are directly associated with a trust value which are further classified into direct trust values, indirect trust values and trust authorization levels. [120] is a work to establish a decentralized trust model. In this, the protocols used to disseminate trust are proposed and the algorithms used to update trust are described. [119] and [120] do not describe how trust policies should be expressed, enforced and integrated with other types of access control policies.

Other models that support behavior trust based on experience and reputation are proposed in [107], [109], [121] and [122]. Alfarez Abdul-Rahman *et al.* [107], [109] discusses a trust model that is grounded in real-world social trust characteristics, and is based on a reputation mechanism. The proposed model allows agents to decide which other agents' opinions they trust more and allows agents to progressively tune their understanding of another agent's subjective recommendations. The goal of this work is to provide a trust model for virtual communities that assist users in identifying trustworthy entities and gives artificial autonomous agents the ability to reason about trust [107], [109]. Some of the other trust models specialize in applying trust for enhancement of resource allocation functions. Farag Azzedin *et al.*[121], [122] present a trust model for grid systems and show how the model can be used to incorporate the security implications into scheduling algorithms. The proposed TRMS (Trust Aware Resource Management System) allocates resources considering a trust relationship between the resource provider (RP) and the resource consumer (RC) [122].

All the models described above are really a key inspiration of our work but these models do not address how to express trust policies and the integration of trust model with authorization framework.

Other proposed security models for grid like [24] and [43] do not deal with trust explicitly and does not provide any trust model to express, validate, update and manage trust relationships. Trust relationships implied from these security models are static and limited. Services like CAS [123] and VOMS [124] address general policy based authorization issues but do not make use of any trust model to express trust policies and to base authorization decisions on it. Similarly the specifications like WS-Trust [35] and WS-Federation [38] provide methods for issuing, renewing and validating security tokens and ways to establish, access the presence of and broker trust relationships but do not give any comprehensive trust model to determine the trustworthiness of different entities in a grid environment.

In the security policy framework, we have made an attempt to define a general trust model that deals with behavior as well as identity trust and discussed its integration with the authorization framework. The model also provides the mechanisms to express, evaluate update and enforce trust relationships and access control policies. It also describes the integration of trust based access with authorization framework.

2.5 Authorization

Authorization, in simple terms, deals with issues like who can access what services/ resources under what conditions. An authorization system can be defined as a system that grants specific type of access to specific requesters based on their authentication, what services/resources they are accessing, current state of the system and their conformation to established authentication, privacy, trust and other security policies. It is a detailed description of all aspects of a system dealing with access of services/resources by requesters. Grids present several unique authorization related challenges. There are several factors that make authorization hard in grid systems *e.g.* user population and resource pool is large and dynamic, resources have different authentication, privacy, trust and authorization requirements, computations span over multiple domains, users have different roles/privileges in different domains *etc.* All these factors make authorization a big and challenging issue. Following paragraph presents the authorization requirements of a grid system.

2.5.1 Authorization Issues

A grid system consists of virtual organizations. Virtual organizations consist of one or more physical organizations or administrative domains. These organizations provide services/resources that can be accessed by subjects of other organizations based on their authorization status. A number of different types of policies can exist among subjects and services in such an environment [125]. If a subject is a member of several domains then his/her access rights can be different in different domains. In order to access a service/resource, the requester must conform to the set of rules/requirements/policies defined by that service/resource. Every administrative domain has its own authentication, privacy, trust and authorization policy and the domain can change it dynamically. So access control in grids need to support multiple security policies and should have the flexibility to support dynamic changes in security policies [126]. In grids, the access depends on many factors like authentication requirements of the service, trust relationship with the requester, privacy requirements of the requester, authorization and other security policies among service providers and requesters *etc.* Authorization in such an

environment should be determined as a result of evaluating the request of an authenticated user against various policies like privacy policy, trust policy, authorization policy *etc*. Many authorization mechanisms for large scale distributed systems like grid and web ignore one of the components from privacy, trust and policy. We emphasize that these are vital components and must be integrated into the access control framework of security architecture to make it more effective and useful [127].

Another desirable feature of authorization mechanism is to support fine grained access to shared resources/services [61], [128]. Support must be there for a flexible delegation mechanism also so that resources/services can be accessed on behalf of a particular user. Scalability and interoperability are also the important requirements that must be addressed by authorization system [48]. The authorization framework needs to be fully distributed also and it should be able to express VO wide, local site wide and other access control security policies related to grid as a whole.

2.5.2 Approaches used for Authorization

There are two general approaches for authorization: identity based and token based [47]. In identity based approach, only the authenticated identity along with the requested action is presented to the resource which then checks an internal list of allowed identities and actions list. If the authenticated identity and requested action appears in the internal list of allowed identities and actions then access is granted otherwise denied. Token based approach is also referred to as capability based authorization. In token based approach, a token (or authorization credential) is granted to the user who then presents it to the service as a proof of his/her rights. The service does not care who the presenter is, rather just that the request came with the appropriate token. The drawback of token based approach is that it is difficult to dynamically revoke access rights whereas in identity based approach, it is difficult to support delegation. Authorization systems can also be categorized as i) pull, push or agent based authorization systems and ii) VO level or resource level authorization systems. Following paragraphs briefly discuss each of these.

Push, Pull and Agent based Authorization

An authorization system where the authorization credentials are pushed to the access controller are called push based authorization systems. The credentials are first obtained

by users from the relevant authority and then these are presented to the system to obtain access to resources. All certificate based systems employ this mechanism. In pull model, the users provide the minimum credentials to the access controller and it is the responsibility of the controller to check the validity of the user based on the policies of the system. The file systems in different operating systems employ pull mechanisms, where users provide their minimum credentials and access policies corresponding to the user are pulled by the operating system. Based on the policies, the operating system either allows or disallows the user to access the resources [48]. Push based mechanisms are more scalable whereas pull based mechanism are more user friendly as users do not have to obtain credentials from the relevant authority. Figure 2.4, Figure 2.5 and Figure 2.6 shows the difference between push, pull and agent based authorization systems.

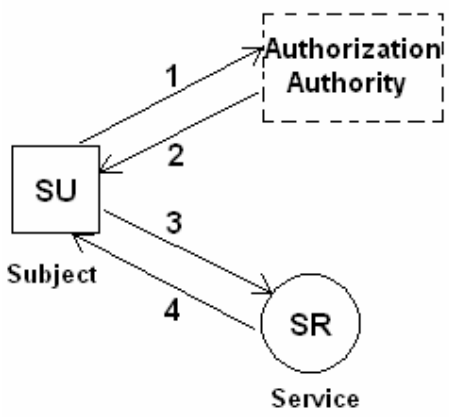

Figure 2.4: Authorization Push Sequence

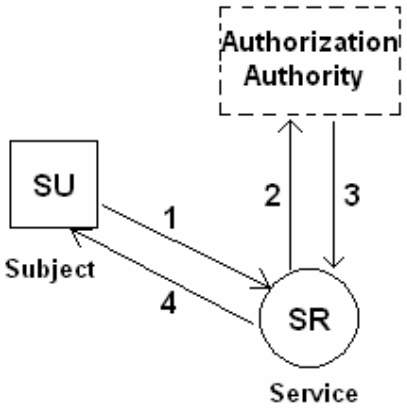

Figure 2.5: Authorization Pull Sequence

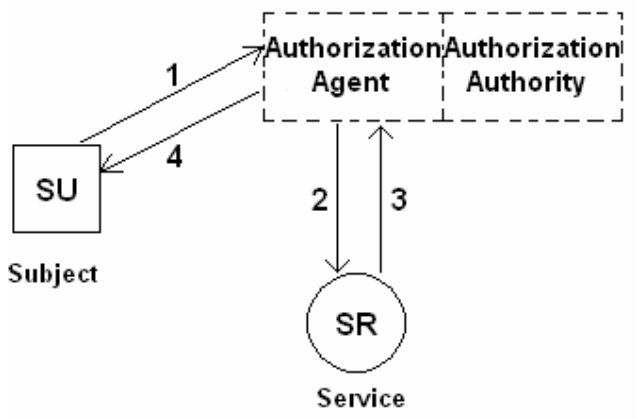

Figure 2.6: Authorization Agent Sequence

In agent based authorization, Subject contacts a higher level agent with a request to obtain service authorization. The agent then makes an authorization decision based on the rules established by authorization authority and if successful, it allows Subject to directly interact with the service [129].

VO Level and Resource Level Systems

VO level grid authorization systems are centralized authorization for an entire virtual organization. These types of systems are necessitated by the presence of a VO which has a set of users, and several resource providers who own the resources to be used by the users of the VO. Whenever a user wants to access certain resources owned by a resource provider, he/she obtains a credential from the authorization system which allows certain rights to the users. The user presents the credentials to the resource provider to gain access to the resource. The Resource Level authorization systems implement the decision to authorize the access to a set of resources. These mechanisms work at the resource level rather than at VO level. VO level and resource level authorization systems look at two different aspects of the grid authorization. These two authorization systems complement each other, and can be implemented together to provide a holistic authorization solution [48].

2.5.3 Authorization in existing middleware

A lot of projects like Akenti [130], CAS [123], PERMIS [90], CARDEA [131], PRIMA [89], GridShib [132], [133], GUMS [134], SESAME [135] and VOMS [124] are

developed to address authorization related issues in one or other form in distributed environments but they have their limitations also. CAS and VOMS are VO level authorization systems whereas Akenti and PERMIS are resource level authorization systems. GUMS is grid user management system. Following paragraphs briefly discuss each of these.

Akenti [130] is an access control mechanism that uses digitally-signed certificates to define and enforce an access policy for a set of distributed resources that have multiple, independent and geographically dispersed stakeholders. The stakeholders assert their access requirements in use-condition certificates. Users are identified by X.509 identity certificates. During a request to use a resource, a policy engine collects all the relevant certificates and decides if the user satisfies all the requirements [130]. Akenti allow different stakeholders to express policies specifying how resources can be accessed but the policy language is different from XACML. Akenti policy is expressed in XML and stored in three types of signed certificates: policy certificates, use-condition certificates and attribute certificates [130]. Thus Akenti policy language model is different from XACML policy language model.

CAS [123] is an approach to the representation, maintenance, and enforcement of policies and provides a scalable mechanism for specifying and enforcing these policies. This approach allows resource providers to delegate some of the authority for maintaining fine-grained access control policies to communities, while still maintaining ultimate control over their resources. The owners of resources grant access to a community account as a whole. The CAS server is responsible for managing the policies that govern access to a community's resources. It maintains fine-grained access control information and grants restricted GSI proxy certificates to the users of community [136]. CAS can also be used for managing role-based sub groups of a virtual organization [137]. CAS also support mechanism to delegate permissions on a set of resources distributed across different administrative domains but it focuses on centralized specification of community policies governing access to collection of resources. Moreover, in CAS, policies are not expressed using XACML.

PERMIS [90] is a policy driven RBAC Privilege Management Infrastructure (PMI). PERMIS implements role based access control (RBAC) scheme in which rights are associated with roles rather than with specific entities. PERMIS describes a policy driven role based access control system. The user's roles and the policy are stored in X.509 Attribute Certificates. The policy, written in XML, describes who is trusted to allocate

roles to users, and what permissions each role has. Access control decisions are made by an Access Control Decision Function. The decision is made according to the requested mode of access, the user's trusted roles and the policy [90]. In PERMIS, policies are written in XML. We are using XACML to express policies which is OASIS standard.

Cardea [131] is a distributed authorization system that facilitates dynamic access control. It is developed as part of the NASA Information Power Grid [138], which dynamically evaluates authorization requests according to a set of relevant characteristics of the resource and requester rather than considering specific local identities. Potentially accessed resources within an administrative domain are protected by local access control policies, specified with the XACML [41] syntax, in terms of requester and resource characteristics. The exact information needed to complete an authorization decision is assessed and collected during the decision process itself. This information is assembled appropriately and presented to the PDP that returns the final authorization decision for the actual access request together with any relevant details. Cardea is currently implemented in the Java language. In Cardea, policies are defined with respect to high level identities such as entity's distinguished name. The authorization decisions depend heavily on the attributes a service requester holds.

PRIMA [89] focuses on the issues of management and enforcement of fine-grained privileges. PRIMA mechanisms enable the use of fine grained access rights and adhoc and dynamic collaboration scenarios. PRIMA provides tools for end users and administrators to manage privileges for the resources they are authoritative for through X.509 Attribute Certificates that carry privilege and policy statements. An access control decision function in the form of an authorization module for the Globus Toolkit® authorizes requests based on the combination of user access privileges with resource policies and provisions low-level access control enforcement functions with decision qualifications [89]. PRIMA system is particularly motivated by the desire to support spontaneous, short lived collaborations among small group of grid users.

GridShib [132], [133] is NSF funded project between NCSA and the University of Chicago. The goal of the project is to integrate GSI [76] and Shibboleth [88] to provide the needed capabilities for a robust attribute infrastructure. This project will deliver a framework that allows participants in multi-organizational collaborations to control the attribute information that they want to publish, share, and reveal to other parties. Those parties will also be able to determine whether they possess the capabilities to access a service by matching their capabilities with the service's shared policy of required

attributes. The pseudonymous interactions will be supported through the use of anonymous public key credentials that are mapped to the client's identity at the client's own discretion [74].

GUMS [134] is another system that automates user registration and management for a local grid site. The model allows users to register once with a VO and provide all the computing sites the information they need with the required level of trust. Tools have been developed to allow sites to automate the management of local accounts and the mapping between grid identities and local accounts [134]. GUMS is not an authorization solution instead it is a user management system.

SESAME [135] is dynamic context-aware access control mechanism for pervasive applications. SESAME complements current authorization mechanisms to dynamically grant and adapt permissions to users based on their current context [135]. The proposed mechanism extends the role based access control (RBAC) model while retaining its advantages. The model dynamically adjusts Role Assignments and Permission Assignments based on context information. Each user is assigned a role subset (by the authority service) from the entire role set. Similarly the resource has permission subsets for each role that will access the resource [135]. SESAME does not make use of any policy language like XACML to express access control policies.

VOMS [124] is a Virtual Organization Membership Service to manage authorization information in Virtual Organization scope. The VOMS System consists of four main components namely User Server, User Client, Administration Client and Administration Server. User Server receives requests from a client and returns information about the user. User Client contacts the server presenting a user's certificate and obtains a list of groups, roles and capabilities of the user. Administration Client is used by the VO administrators for adding users, creating new groups, changing roles *etc.* Administration Server accepts the requests from the clients and updates the database. The VOMS architecture uses the authentication and delegation mechanisms provided by the Globus Toolkit® Grid Security Infrastructure (GSI) [76], [124]. VOMS constitutes a system conceptually similar to CAS. It has a community centric attribute server that issues authorization attributes to members of the community.

There are many other attempts also like [139] - [156]. These approaches are mainly concerned with policy based authorization and access control but do not provide separate mechanisms for dealing with privacy and trust based access control. Xinwen Zhang *et al.* [139], [140] propose a usage control (UCON) based authorization framework for

collaborative applications. Usage control policies are defined using subject and object attributes along with system attributes as conditions. The conditions can be used to support context based authorizations but the framework does not support obligations. Christian Schlager *et al.* [101] proposes authorization framework respecting privacy and flexibility in b2c eCommerce but it does not support context based authorization. The framework proposed by Jin Wu *et al.* [141] mainly supports fine grained access control for resource usage and management. Ludwiz Seitz *et al.* [142] present a system primarily enabling delegation using XACML access control system. Hai Jin *et al.* [136] proposes a universal, scalable authorization and access control framework called RB-GACA for grid systems but the framework does not provide facilities to integrate different authorization mechanisms. It is based on role based access control mechanism. Glenn Wasson *et al.* [143], [144] and Jun Feng [145] mainly focus on virtual organization wide resource usage and access control policy expression and enforcement mechanisms. Efforts like [146], [147], [148], [149] and [150] have their own mechanisms to express, evaluate and enforce access control policies and do not use XACML. The implemented authorization model is distinguishable from the discussed projects/models in different ways *e.g.* the implemented model supports fine grained and context based access. Policy expression is platform independent. The framework is able to express and enforce VO wide and service wide access control policies. It extends basic authorization mechanism to include trust and privacy based access to grid services. It also supports multiple security policies and provides facilities to integrate different authorization mechanisms.

Table 2.1 compares the existing middleware in terms of their support in the areas of authentication, privacy, trust and authorization.

Middleware / Projects	**Authentication**	**Privacy**	**Trust**	**Authorization**
CredEx	√	X	X	X
Entrust TruePass	√	X	X	X
Kerberized Certificate Authority	√	X	X	X
Liberty Alliance Project	√	X	X	X
Microsoft .NET Passport system	√	X	X	X
Microsoft Trust Bridge	√	X	X	X
MyProxy	√	X	X	X
Password Safe	√	X	X	X

EPAL	X	√	X	X
P3P	X	√	X	X
KeyNote	X	X	√	X
PolicyMaker	X	X	√	X
PeerTrust	X	X	√	X
XenoTrust	X	X	√	X
NICE Framework	X	X	√	X
SeGO	X	X	√	X
TrustBuilder	X	X	√	X
Akenti	X	X	X	√
CARDEA	X	X	X	√
CAS	X	X	X	√
PERMIS	X	X	X	√
PRIMA	X	X	X	√
SESAME	X	X	X	√
VOMS	X	X	X	√
Shibboleth	√	√	X	√
GridShib	√	√	X	√
CRISIS	√	X	X	√
Legion	√	X	X	√
GT4	√	X	X	√

Table 2.1: Comparison of existing middleware in terms of their support in the areas of authentication, privacy, trust and authorization.

In Table 2.1, middleware are marked under the column (authentication, privacy, trust and authorization) for which they provide the major support/system. This does not mean that these middleware cannot be used in conjunction with other middleware/mechanisms/technologies/systems to address other issues. *e.g.* Akenti has been marked under authorization column because it proposes and describes a specific authorization model and policy language. It has not been marked under authentication, though it support authentication through digitally signed certificates, because it does not proposes any new authentication model. Similarly other middleware like PERMIS, PRIMA and SESAME can authenticate users but marked under authorization column because more precisely these are authorization systems.

2.6 Problem Formulation

As observed during exhaustive review carried out in the previous section and summarized in Table 2.1, it is clear that grid services are continuously evolving, defining complex security requirements and policies between service providers and consumers. The multiplicity and complexity of security requirements and policies demand a security policy framework. Access to resources/services in a grid is provided based on conformance to established authentication, privacy, trust and authorization policies. These policies differ in their type, nature and scope. Most of the security solutions available do not provide comprehensive models to address different types of security requirements and access control policies. Many of them are not based on open standards. Some of them are problem specific or use proprietary mechanisms. As per Table 2.1, no middleware for grid services exist that integrates privacy, trust and policy based access control in a single authorization framework. Most of the existing middleware address one of the issues from authentication, privacy, trust or authorization and do not attempt to integrate all these in one common security framework. So a general, standards based integrated security policy framework is essentially required that addresses key security requirements related to authentication, privacy, trust and policy based authorization, and provides support to express, evaluate and enforce different types of access control policies in a uniform, integrated and platform neutral way. The integrated model will enable us to treat and specify the scenarios/issues concerning combination of authentication, privacy, trust and authorization as a single unit. Currently, there are very few initiatives towards defining such a framework. This thesis makes an attempt to define and implement such a framework that addresses important security requirements related to authentication, privacy, trust, authorization, and provide support to express, evaluate and enforce security policies related to them.

2.7 Objectives of the thesis

The first objective is to study and explore grid and web services to acquire basic knowledge about their manner of operation, execution, applications and differences between them. The second objective is to identify and categorize different types of

security requirements and policies and to describe how security policies should be expressed and exposed. In the implemented framework, we have categorized security requirements and policies under four heads namely authentication, privacy, trust and authorization. Third objective is to define an integrated security policy framework for grid services. By security policy framework we mean the security architecture that addresses key security requirements of authentication, privacy, trust and authorization, and provide support to express, evaluate and enforce policies related to these requirements. The framework provides a set of features and services that tackle a set of security requirements and scenarios related to authentication, privacy, trust and authorization. Fourth and the last objective is to demonstrate the use and applicability of the framework.

To achieve the first objective, a comprehensive study of grid and web services architectures, their manner of operation, execution, applications and differences among them has been done. A thorough study of standards and specifications used in grid and web services based systems has also been carried out. Work done in the area of authentication, privacy, trust and policy based access in grid systems has also been studied in detail.

To achieve the second objective, the key security requirements and policies of grid systems have been categorized under four major heads namely authentication, privacy, trust and policy based authorization. Then the approach used for expressing and exposing these policies in the environment is discussed.

To achieve the third objective, we have implemented models for authentication, privacy, trust and policy based authorization and integrated them. Authentication Model provides support for single sign-on and delegation features through the use of proxy certificates. It also presents a credential management service to store, retrieve and update multiple user credentials. The privacy model provides support for anonymous access, hidden service access and access to private information based on conformance to privacy policies. The trust model provides support for calculating direct as well as recommended trust to determine trustworthiness of target service/resource. The policy based authorization model provides policy based access to grid services. The detailed explanation of these models and their integration is presented in the fifth chapter.

To achieve the fourth objective, the framework has been evaluated by implementing different security related scenarios and through implementations that involve enforcement

of different types of access control policies. The implementation has been done in .NET environment with the support of WSE 3.0 toolkit.

Chapter 3

Elements identified for the Framework

This chapter explains the different elements/entities identified and defined in this work in order to understand the security policy framework. The security policy framework and its authentication, privacy, trust and authorization models are based on these elements. Following are the major elements/entities of the framework.

Subject (SU): Subject is an entity that wants to access services/resources. It can be a user, a service or any other entity on behalf of user/service.

Service (SR): Service is a piece of software that provides some functionality and can be accessed by Subjects or other Services. Services are exposed in the environment along with their associated Service Policies and are found by Subjects. Services are provided by different Service Providers.

Resource (R): Resource is a sharable object that can be accessed by Subjects. It can be a CPU, a storage device, software, data, scientific instrument or any other peripheral. Subjects access Resources through Services. In other words, a Resource provides a Service. Subjects can access Resources based on their authorization status and their conformance to established authentication, privacy, trust, authorization and other security policies. There are two types of resources:

i) Subject's Resources: These are the resources which are provided by Subjects to Service Providers. *e.g.* Subject's name, date of birth, his private telephone number *etc.*

ii) Service Provider's Resources: These are the resources which are provided by Service Providers to Subjects. Subjects can access these Resources based on their authorization status and their conformance to established security policies.

Service Policy (SrP): Service Policy refers to the set of rules/requirements associated with a Service. A Subject must conform to Service Policy in order to access that Service.

Domain (DO): Domain refers to the set of Subjects, Services and Service Providers under a unique Domain Policy (DP). The Services in a Domain are provided by Service Providers and they may belong to same or different physical organizations/institutions.

Domain Policy (DP): Domain Policy refers to the set of rules/regulations/requirements of a Domain, which a Subject must satisfy in order to access the Services/Resources provided by that Domain.

Service Provider (SP): The term Service Providers refers to physical organizations/ institutions that provide Services/Resources in a Domain. A Service Provider can provide Services/Resources in any number of Domains. Services/Resources in a Domain may come from same or different Service Providers. Thus a Domain is a more dynamic entity than individual physical organization/institution.

Service Provider Policy (SPP): The Service Provider Policy refers to the set of rules/regulations/requirements of a Service Provider, which a Subject must satisfy in order to access the Services/Resources provided by that Service Provider.

Access (AC): Access is an operation or set of operations that a Subject performs on a Service/Resource. The access is provided based on conformance to Service Policy (SrP) and other applicable security policies that are associated with that Service/Resource or the access request.

Filter (FL): The rights/privileges of a Subject are different in different Domains. Filter is a component through which rights/privileges of a Subject are attenuated / filtered for a particular Domain. There are two types of Filters: Filter-In (FL-I) and Filter-Out (FL-O). Through Filter-Out component, the Subject leaves the Domain with access rights that his Domain grants to him. Through Filter-In component, the Subject enters the target Domain with access rights that the target Domain grants to Subject's Domain. These will be explained in detail later in this section.

Policy (PO): Policy is a set of rules/requirements that can be associated with Subject/Service/Service Provider/Domain *etc.* Policies can be of different types. Policy can be represented as set PO = {ANP, PP, TP, AP, OP} where

ANP is Authentication Policy

PP is Privacy Policy

TP is Trust Policy

AP is Authorization Policy

and OP refers to any Other Policy

These policies deal with issues specific to authentication, privacy, trust, authorization and other aspects and will be defined and explained in Chapter 4. Service Policy (SrP), Service Provider Policy (SPP) and Domain Policy (DP) are subsets of Policy (PO). *i.e.* SrP $\subseteq$ PO, SPP $\subseteq$ PO and DP $\subseteq$ PO.

Privacy Index (PI): Privacy Index indicates the privacy level of Subject's Resources. Subject's Resources are marked with a PI value. Higher values of PI mean more privacy. Private data/information is marked with higher values of PI. PI can take following values:

PI Value	Privacy Level	Meaning
0	No Privacy	Subject's Resource can be used anyway.
1	Partial Privacy	Subject's Resource can be used with permission only.
2	Time Limited Privacy	Subject's Resource can be used with permission but for a limited time period.
3	Full Privacy	Subject's Resource can not be used under any circumstances.

Table 3.1: Privacy Index levels and their meaning

Purpose (PU): Purpose element tells the purpose for which Subject's Resources (*e.g.* his private/public information) are required by Service Providers. A Service Provider may require Subject's Resources for research / marketing / data mining / to provide customized access to Services *etc.*

Action (ACT): Action element tells what operations/actions can be performed on i) Subject's Resources by Service Providers and ii) Service Providers' Resources by Subjects. These operations/actions can be read/ write/ execute/ copy/ share/ distribute/ update/ delete/ append/ view *etc.* Exact operation depends on the resource under consideration. Different types of Resources support different types of operations/actions on themselves.

Conditions (CO): Conditions are privacy statements which describe some prerequisites that must be satisfied before granting access to Resources/Services. *e.g.* a Condition may state that Subject's age must be greater than 18 in order to access a Service/Resource. Though Conditions can be specified through policy associated with that Service/Resource but specifying privacy statements separately for a Service/Resource

enable us to implement privacy based access control in a more meaningful and efficient way. This will be explained later in the Privacy Model.

Obligations (OB): Obligations are activities that must be performed by Service Providers while providing access to Resources/Services or while accessing Subjects Resources. Example of obligation can be to send a notification e-mail to parents if their children access a particular Service.

Consent (C): Consent is the permission given by Subjects/Service Providers, who provide their Resources, to explicitly state that their Resources/Services can be used for a particular Purpose. *e.g.* a Subject may explicitly state that his telephone number can be used by marketing people to inform him about new schemes.

Authority (A): Authority is an administrative entity that is capable and authoritative of issuing, validating and revoking an electronic means of proof. Authority can be classified as i) Identity Authority ii) Authorization Authority and iii) Attribute Authority

i) Identity Authority: Identity Authorities make assertions about the identity of the Subject. Certificate Authorities (CAs) are examples of Identity Authorities.

ii) Authorization Authority: Authorization Authorities make assertions about the authorization rights of a Subject. In the implemented framework, Filter components make authorization assertions. Identity Authorities enable authentication whereas Authorization Authorities enable authorization.

iii) Attribute Authority: Attribute Authorities issue attribute assertions that a given Subject has one or more attribute/value pair.

Privacy Controller (PC): is an entity that checks misuse of Subject's Resources at Service Provider's end *i.e.* it works for Subjects interests. It makes sure for Subjects that Service Providers are respecting Subject's privacy requirements and are accessing/ providing access to Subject's Resources only for the purpose for which they have been sent.

Trusted Third Party (TTP): is an independent entity that is trusted by both, the Service Requesters (Subjects) and Service Providers. Subjects can use it to obtain signed credentials which they can later provide to Service Providers to obtain access (including anonymous access) to their Resources/Services. Service Providers can make use of TTP to hide Service details from Subjects. TTP plays an important role in handling Service Provider's and Service Requester's privacy requirements. TTP can be run by a government agency in order to make Service Providers and requesters to trust it. PC can also be a part of TTP.

MAP (MAP): is an operation that maps/transforms Subject of one administrative domain to Subject in another administrative domain. *e.g.* SU_i (DO_k) → map→ SU_j (DO_l) means that Subject SU_i of Domain DO_k has been mapped to Subject SU_j in Domain DO_l.

Grid Elements Set (GES): refers to the set GES = {SU, SR, R, SP, DO, A, PC, TTP} *i.e.* the set of entities that interact with each other in a grid environment.

Grid Environment (GE): refers to the distributed environment in which different entities of Domain or sets of different Domains interact with each other.

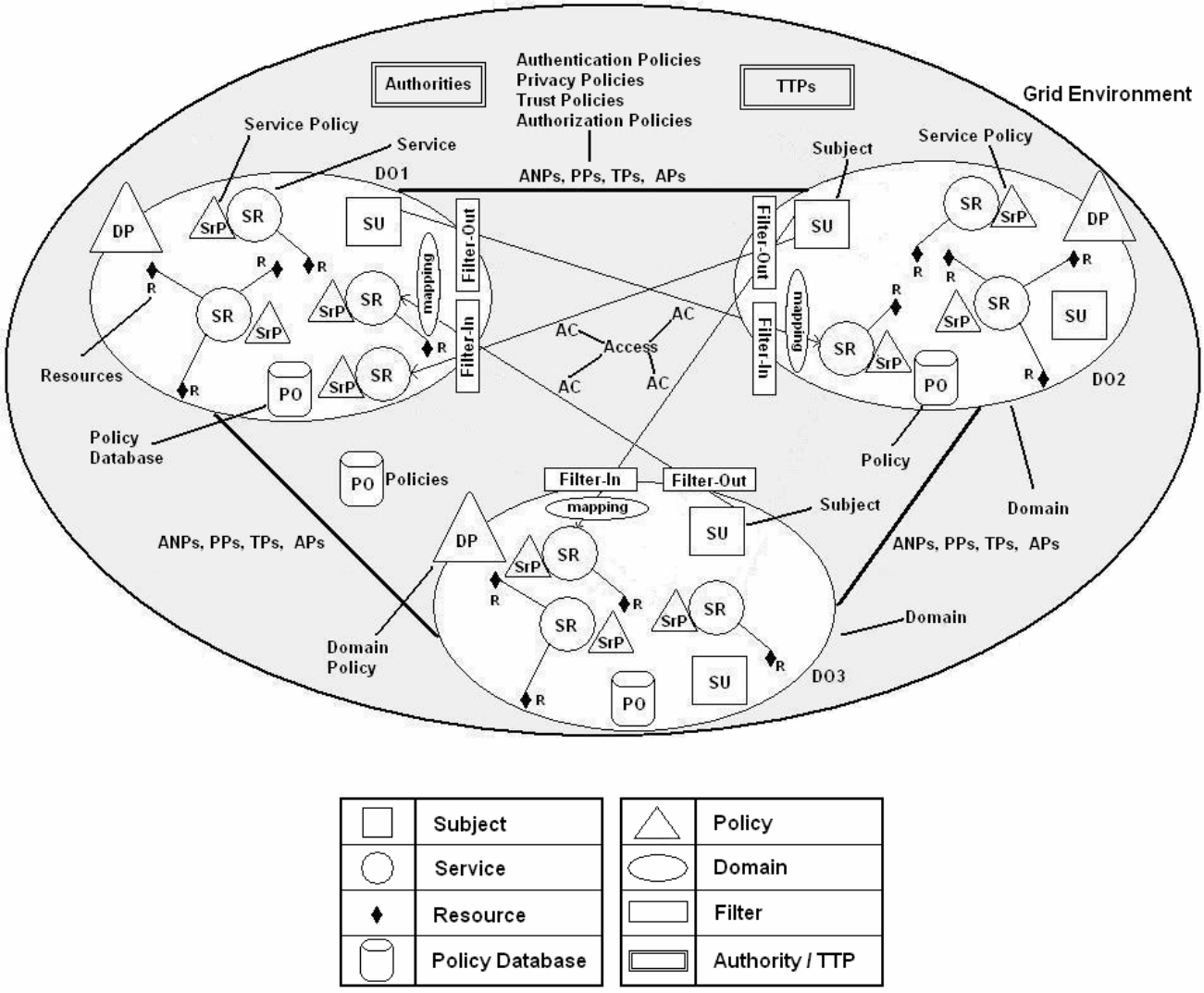

Figure 3.1: Schematic showing Grid Environment consisting of three domains along with other elements of the security framework

In a typical Grid Environment, the grid entities described above interact in a complex manner. The secure access of Services/Resources among different entities in such an environment demands a security framework. The focus of the thesis is to implement such a framework.

Figure 3.1 represents a blueprint of the envisioned Grid Environment. It consists of three Domains (DO1, DO2 and DO3) along with other elements of the security framework. In the diagram squares represent Subjects, circles represent Services, diamonds represent Resources, triangles represent Policies, rectangles represent Filters and ellipses represent Domains. As shown in Figure 3.1, Subject's access request for Service SR first passes through Filter-Out component at the source Domain and then through Filter-In component at the target Domain. During this passage, Subject's access rights are filtered for the target Domain. Through Filter-Out component, the Subject leaves the Domain with access rights that his parent Domain grants to him and through Filter-In component, the Subject enters the target Domain with access rights that the target Domain grants to Subject's parent Domain. In other words, the Subject gets the intersection of the rights that his parent Domain grants to him and the rights that target Domain grants to Subject's parent Domain. Filter components make the authorization system scalable and flexible.

To understand the use of Filter-In and Filter-Out components consider that a Domain, say Domain-A, has 100 subjects who perform either "read", "update" or "execute" operations on a file through Service SR-1 provided by Domain-B. As the number of subjects accessing the Service SR-1 is large, it will be difficult for Domain-B to maintain access rights information of each and every subject of Domain-A, for SR-1. So here Domain-B will store the information that any of the subjects from Domain-A can perform read, update and execute operation on file through SR-1, but, the more fine grained access rights information *i.e.* which subjects of Domain-A can perform which actions on that file through SR-1, will be maintained by Domain-A itself. This concept will make the authorization system flexible and scalable. This concept has been implemented in the framework through Filter-Out and Filter-In components. In this particular example, Filter-Out component will store access rights information regarding which subjects of Domain-A can perform which actions on SR-1 and Filter-In component will store access rights information regarding the actions that can be performed on SR-1 by the subjects of Domain-A.

Now consider that Domain-A allows one of its subjects, say SU-1, to perform "read" and "update" operations on the file through SR-1. If SU-1 tries to perform "execute" operation on the same file then Filter-Out component will not allow this request to pass through it as it has information regarding which subject can perform which action on file. As SU-1 is not allowed by Domain-A to perform "execute" operation on file, Filter-Out component will restrict this access request. On the other side, if Domain-A allows one of its subjects to perform "delete" operation (intentionally or unintentionally) on the same file then Filter-In component of Domain-B will not allow this request to pass through it as it has information regarding the actions that can be performed on SR-1 by the subjects of Domain-A. As Domain-B does not allow any of the subjects of Domain-A to perform "delete" operation on the file, Filter-In component will restrict this request. Filter components are making the authorization system scalable *e.g.* in this case, subject wise access rights information is being maintained by Domain-A through Filter-Out component and domain wide access rights information is being maintained by Domain-B through Filter-In component. A single Domain is not maintaining all the access rights information. Thus the authorization system is scalable.

The mapping operation (MAP) is performed to provide the Subject an identity that is local to the target Domain. The mapped identity is used by the target Domain to provide access to the requested Service/Resource. The mapping operation is optional. It is performed only if access of a Service/Resource explicitly requires local identity. If the access of a Service/Resource does not require local identity then MAP is not performed. The role of other elements like Privacy Index, Purpose, Privacy Controller etc. will be explained in Chapter 5 while explaining Privacy Model as these elements play major role in providing privacy based access to grid services.

Figure 3.1 also shows Policy database to store different types of policies. Domains can have different authentication, privacy, trust and authorization related requirements and policies among each other. These policies can also exist among Subjects and Services of different Domains in a complex manner. Determining whether a Subject conforms to all applicable policies is a complex task. In the implemented framework, it is performed by Authorization Handler with the help of other components like Privacy Handler, Trust Handler etc. At the target Domain, the integrated policy based authorization framework checks Subject's conformance to Domain Policy (DP), Service Provider Policy (SPP), Service Policy (SrP) and other applicable policies. If Subject does not conform to any of these policies, the access is denied.

The detailed architecture and explanation of the implemented framework is given in Chapter 5. Next chapter describes different types of security policies that have been identified, categorized and focused in the framework. The chapter also describes how these policies have been expressed and exposed in the environment.

Chapter 4

Security Policies Categorization for the Framework

This chapter categorizes and explains different types of security policies that have been focused in the implemented framework. The chapter also illustrates the methods for expressing and exposing these policies in the environment.

4.1 Security Policies

The definition of the term policy is often contextual and confusing. Policies are defined, implemented and utilized in a particular context. There can be policies for security, workload, quality of service, networking service, business processes and a multitude of other areas. Generally policies are classified under three levels: business level, domain level and device level [6]. Business level policies are concerned with high level business definitions, domain level policies are concerned with organization level issues and device level policies are concerned with issues related to individual resources. The implemented framework is mainly concerned with security policies that control access to grid services/resources. Some of these policies will operate at domain level and others will operate at device level. This chapter discusses different types of security policies that can exist among different entities in a grid environment. These policies operate at different levels of abstraction and the mechanisms for their expression, evaluation and enforcement are also different. The different policies have been classified into two categories:

i) Access Control Policies
ii) Non-Access Control Policies

Following paragraphs briefly discuss each of these.

4.1.1 Access Control Policies

Access control policies are those policies which play direct role in determining whether a resource/service can be accessed or not. Access control policies are mainly attached with services/resources but can also be applicable to domains and service providers. *e.g.* a domain may have the policy that trust with the requester must be greater than a particular threshold value in order to access any of the services/resources provided by that domain. Similarly a service provider may also have the policy that all the requesters must have X.509 certificates in order to access any of his services/resources. These policies are the example of access control policies applicable to a domain and a service provider. On the other side consider the policies: i) Service "SR-1" is accessible only to subjects of Domain "D" on Mondays from 09:00 am to 05:00 pm only and ii) Any subject of any domain can access the service "SR-2" but only for research purpose. These are the examples of access control policies applicable to a service. The focus of the thesis is on to describe mechanisms to express, evaluate and enforce access control policies associated with services/resources, domains and service providers. Different resources/services can have different number of access control policies associated with them. A subject must satisfy these policies in order to access that service/resource. The access control policies have further been classified into authentication policies, privacy policies, trust policies and authorization policies.

4.1.2 Non Access Control Policies

Non access control policies do not play direct role in determining access to a resource /service rather their nature and scope is different. *e.g.* a policy might state that if subject "S" of domain "D" accesses the service then data generated during access will be stored on hard disk "HD1" but for all other subjects it will be stored on hard disk "HD2". This policy is different from access control policy as enforcement of this policy comes into picture only once we have determined that the service/resource can be accessed. So it cannot be considered as access control policy in true sense. A non access control policy might also state that execution of a grid service cannot take more than one hour on any

machine for subjects of domain “E”. Similarly the policies like: “remove machine “M” from the grid if utilization falls below 50%” and “data stored on hard disk “HD4” will be replicated on hard disks “HD9” and “HD10” on alternate days”, are examples of non access control policies. Compared to access control policies, these policies are different in nature, scope and cover resource usage, operational and business level issues of virtual organizations. These types of policies must be enforced for all applicable access requests which are granted access to the requested service/resource. The mechanisms to express, evaluate and enforce these policies are very much different from those of access control policies. The implemented framework is mainly concerned with access control policies.

4.2 Access Control Policies Categorization

The access control policies have been categorized into authentication related, privacy related, trust related and authorization related policies. Figure 4.1 shows where among different grid entities in a grid environment, the authentication, privacy, trust, and authorization related access control policies can exist.

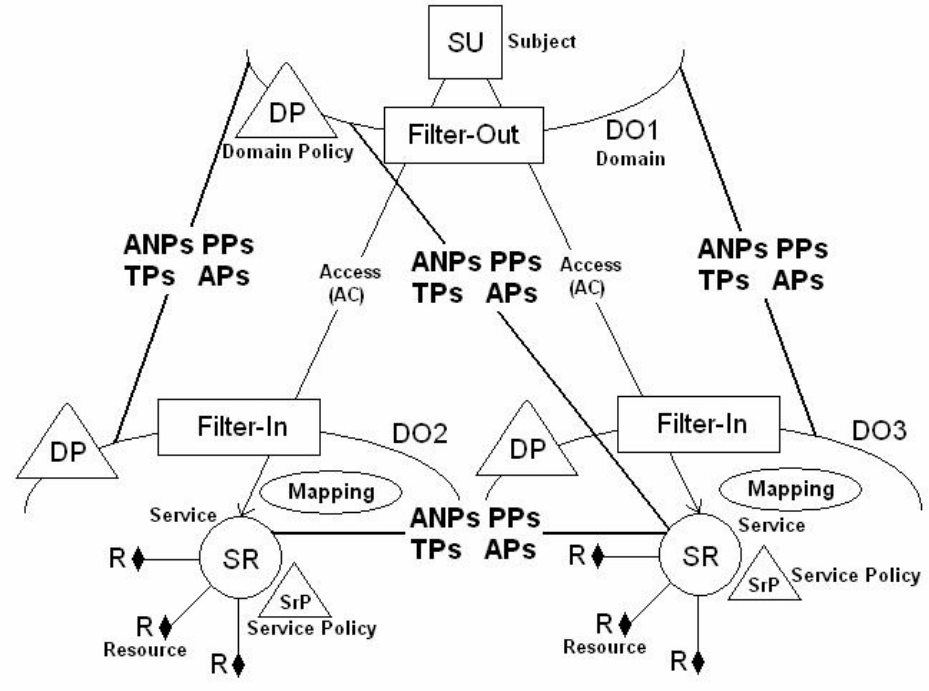

Figure 4.1: Schematic showing authentication, privacy, trust and authorization policies among different entities in a grid environment

In Figure 4.1, SU represents Subject, DO represents Domain, ANP represents Authentication Policy, AP represents Authorization Policy, TP represents Trust Policy, PP represents Privacy Policy, SR represents Service, SrP represents Service Policy and R represents Resource. Every service/resource is associated with Service Policy (SrP), which must be satisfied in order to gain access to that service/resource. SrP may include ANPs, PPs, TPs and APs. In general, ANPs, PPs, TPs and APs can exist among DO↔DO/SP/SU/SR, SP↔SP/SU/SR, SU/SR↔SU/SR and access to SR/R is controlled by SrP. Following sections discuss each of these access control policies.

4.2.1 Authentication Policies (ANPs)

Authentication policies deal with authentication related issues like authentication mechanisms and type of security credentials required by subjects for the invocation of a grid service. Based on authentication policies published by a service, users can take necessary measures to conform themselves to these policies. A grid service may require a particular authentication mechanism for its invocation. In grid environment, authentication mechanisms employed by service providers of different domains can be different. *e.g.* one service provider can use X.509 certificates and others can use Kerberos / username: password / custom security tokens. The services can also demand different security credentials from subjects of different domains based on their roles/trust relationships. *e.g.* a service can demand X.509 certificate from subjects of domain "A" and Kerberos ticket from all other subjects. The authentication policy can also specify whether the service accepts proxy credentials, delegated credentials, custom security tokens or anonymous security tokens *etc.* and if supported then under which conditions or with what constraints? *e.g.* if a service supports single sign-on through proxy certificates then up to which level they are acceptable? A critical service may decide not to accept proxy / delegated credentials from subjects. Using authentication policy, a service can also convey encryption and signature requirements. In grid, services should be able to convey the required authentication mechanisms / algorithms / requirements to different subjects so that subjects can access it. This can be achieved by publishing the authentication policy. Thus mechanisms to express, evaluate and enforce authentication policies are essentially required.

4.2.2 Privacy Policies (PPs)

Privacy policies describe privacy related concerns among service providers and requesters. Service providers may want a service to be accessed only for a particular purpose. They may also want a service to be accessible for a particular purpose by a particular subject with particular credentials. They may also want a service to be accessed for one purpose by subjects of one domain and for another purpose by subjects of another domain. All these are the examples of privacy policies on services. Mechanisms should be there to describe and enforce privacy policies among services and subjects of same/different domains.

Using privacy policies service requesters can also specify the purpose for which they want their private data / information to be accessed. Service requesters can also tell service providers as how they want their personal data / information to be used for a particular purpose. These types of privacy requirements are stored in the database in the form of privacy relationships. Representation of privacy relationships is explained in Section 5.2.1 The privacy relationships are used by authorization framework to provide privacy based access to services/resources.

In the implemented framework, privacy policies applicable to a service/resource have been attached with it through Service Policy (SrP). A subject must conform to privacy policy associated with the service in order to access that service and a service must also respect the privacy policies of a subject if it exposes subject's private data / information to other entities.

4.2.3 Trust Policies (TPs)

Trust policies describe trust related concerns of service providers and requesters. *e.g.* A grid domain may want X.509 certificates to be signed by a particular CA from members of one domain and may not want to accept the same from members of another domain. Subjects may want to access a particular service/resource if the direct or recommended trust with target service/service provider/domain is greater than a specific limit and not otherwise. A service also may require different trustworthiness from subjects of different domains for its access. A service may also want to accept delegated credentials if trust with the subject is greater than a threshold value and not otherwise. All these scenarios

involve trust related concerns and are examples of trust policies. Service providers and requesters may also want to enforce different trust relationships for different contexts. So mechanisms should be there to describe trust relationships and policies among subjects and services of different domains. The representation of trust relationships is explained in Section 5.3.1 while describing trust model.

Like privacy policies, trust policies applicable to a service/resource are associated with it through Service Policy (SrP). A subject must conform to these policies also in order to access that service. The implemented framework makes use of established trust relationships and policies to provide trust based access to grid services/resources.

4.2.4 Authorization Policies (APs)

Authorization policies deal with issues like who can access which services/resources and under what conditions. Access to a particular service/resource is controlled by set of authorization policies (along with authentication, privacy, trust and other types of policies) that are enforced at service provider's end. Authorization policies can be very complex as they may include a number of different types of constraints and conditions to be satisfied before granting access to the requested service/resource. Authorization also depends on the attributes held by subject, service/resource or environment. Authorization policies describe rules regarding which subjects can access which services/resources, when they can access the services/resources, under what conditions they can access the services/resources, what actions they can perform on the services/resources *etc.* Authorization policies may state whether all subjects or only the subjects belonging to a particular domain or specific subjects from specific domains can access the services/resources. Authorization policies applicable to a particular service/resource can be different for subjects of different domains. Authorization policies may also vary with the level of trust and privacy relationships with the target domain/service provider.

The access control policies described above (authentication, privacy, trust and authorization) are related with each other in a complex manner. *e.g.* Trust policies may include privacy policies. Authorization policies may require access to be provided if appropriate privacy and trust policies are enforced. Similarly, authentication may also be governed by policies directed by trust model. A single policy may include different

aspects *e.g.* the policy "Resource R1 is accessible through X.509 certificate from 10 am to 11 am on Wednesday; only for the research purpose and only if the trustworthiness of the subject is greater than 0.8; and if the subject is researcher of ABC domain." describes that authentication, privacy, trust and authorization related aspects can be present in a single policy.

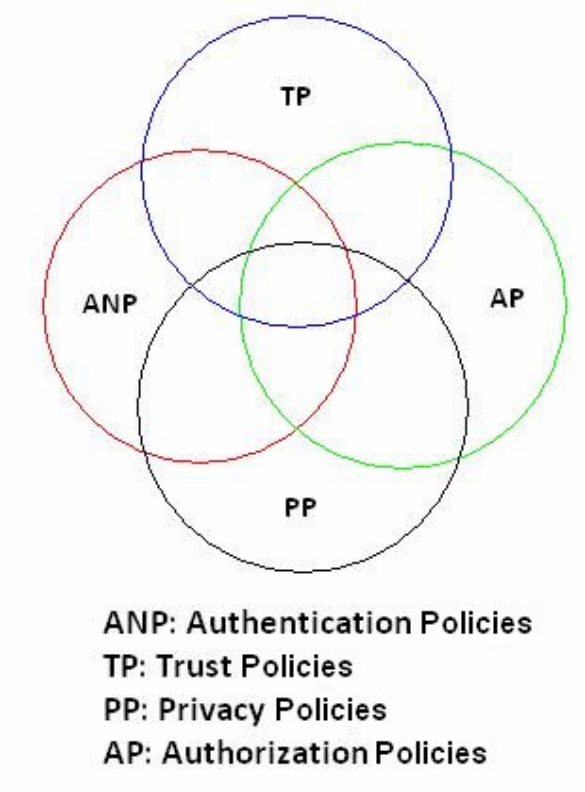

Figure 4.2: Interrelationship among different types of policies

Figure 4.2 gives an indication that authentication, privacy, trust and authorization related aspects can overlap and different combinations of these aspects are possible. The integrated model is an attempt to treat a policy involving these combinations as a unit. The policy specification and evaluation related functionality of authentication, privacy and trust related aspects have been incorporated into integrated policy based authorization model. The integrated model described in Chapter 5 is capable of handling a policy involving different combination of authentication, privacy, trust and authorization related aspects as a single unit. Without integrated model, it is not possible to treat the policy involving combination of authentication, privacy, trust and authorization related aspects as a single unit.

All types of access control policies discussed above are applicable to services/resources, domains and service providers but the mechanisms to associate these policies to services/resources, domains and service-providers are different. The policies applicable to services/resources are associated with them through Service Policy (SrP) and the policies applicable to domains and service providers are associated with them through Domain Policy (DP) and Service Provider Policy (SPP). Next section describes how these policies have been expressed in the security policy framework.

4.3 Policy Expression

In the security policy framework, the access control policies related to authentication, privacy, trust and authorization have been expressed in XACML and are stored in the policy database maintained by every domain. The main reason behind using XACML is that it is the most promising and notable effort to express and enforce general access control policies. It is OASIS standard and provides an extendable and portable way to express access control policies. XACML allow different domains to express and exchange policy information in an interoperable way. To express unique features of privacy and authorization policies, we are making extended use of XACML.

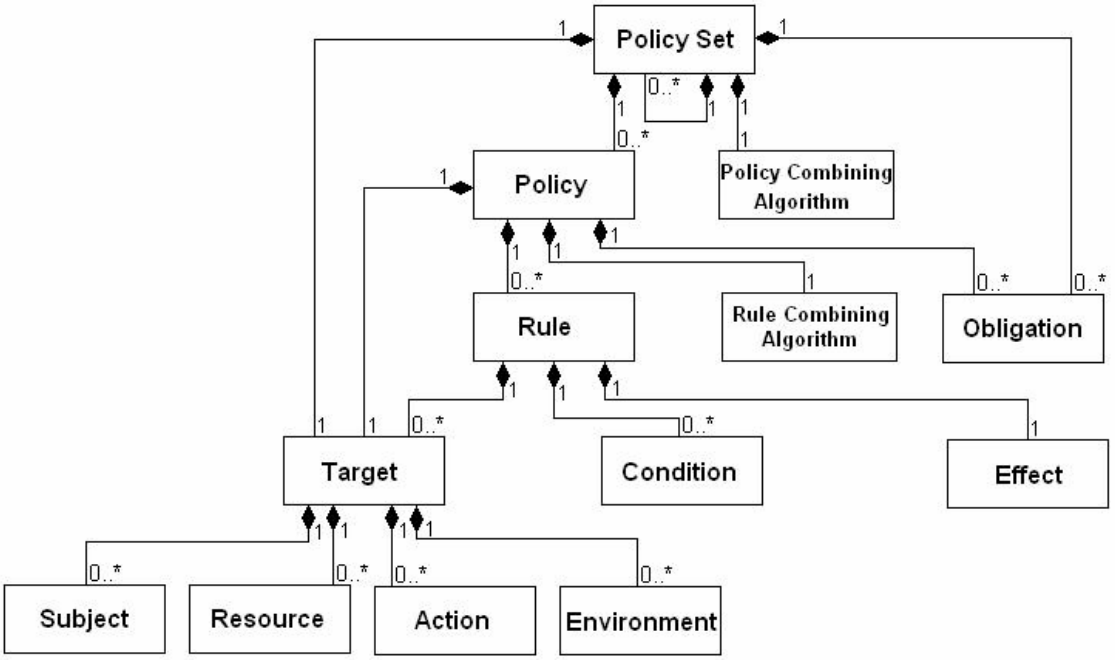

Figure 4.3: XACML policy language model

In XACML, policy is constructed as a set of rules against the target defined as a triod (Subject, Resource, Action). Figure 4.3 shows XACML policy language model and the relationship of its different elements with each other. The main elements of policy language model are: Rule, Policy and Policy Set. Following paragraphs briefly discuss each of these elements.

A Rule is the most elementary unit of policy. A Rule is a defined as "A target, an effect and a set of conditions". A target is defined as the space of decision requests that refer to actions on resources by subjects. Target helps in determining whether or not a rule is relevant for a request. An effect is either permit or deny, and is the intended consequence of the satisfied rule. Conditions are statements about attributes that upon evaluation returns either true, false or indeterminate. The conditions involve the calculation of attributes of the subject, the resource, the action or the environment. Conditions make the rule dynamic [41]. Multiple rules can be associated to a policy.

A Policy consists of a set of rules, a rule combining algorithm, a target and a set of obligations. A rule combining algorithm specifies the procedure by which the results of evaluating the component rules are combined when evaluating the policy. Obligations are the actions that must be performed by the enforcement system in conjunction with the enforcement of an authorization decision. A Policy Set consists of a set of policies, a policy combining algorithm, a target and a set of obligations. A policy combining algorithm specifies the procedure by which the results of evaluating the component policies are combined when evaluating the policy set. The obligations are the operations specified in the policy set that should be performed by the enforcement system in conjunction with the enforcement of an authorization decision [41].

XACML specification also defines six rule-combining and policy-combining algorithms namely: Deny-overrides, Ordered-deny-overrides, Permit-overrides, Ordered-permit-overrides, First-applicable and Only-one-applicable. In Deny-overrides, if any rule evaluates to deny, then the final authorization decision is also deny. Ordered-deny-overrides is same as Deny-overrides, except the order in which relevant rules are evaluated is the same as the order in which they are added in the policy. In Permit-overrides, if any rule evaluates to permit, then the final authorization decision is also permit. Ordered-permit-overrides is same as Permit-overrides, except the order in which relevant rules are evaluated is the same as the order in which they are added in the policy. In First-applicable, the result of the first relevant rule encountered is the final authorization decision. In Only-one-applicable, the result of the only one applicable

policy is the final authorization decision. Figure 4.4 shows the skeleton of an access control policy written in XACML.

```
⋮
<Policy ... PolicyId = "" ... RuleCombiningAlgoId= " " ...   >
      ⋮
      <Target>
      ⋮
      </Target>
      ⋮
      <Rule ... RuleId = "" ... Effect = "" ... >
            ⋮
            <Target>
                  ⋮
                  <Subject>
                  ⋮
                  </Subject>
                  ⋮
                  <Resource>
                  ⋮
                  </Resource>
                  ⋮
                  <Action>
                  ⋮
                  </Action>
                  ⋮
                  <Environment>
                  ⋮
                  </Environment>
                  ⋮
            </Target>
            ⋮
            <Condition>
                  ⋮
            </Condition>
                  ⋮
      </Rule>
      ⋮
      <Obligation>
            ⋮
      </Obligation>
      ⋮
</Policy>
⋮
```

Figure 4.4: Skeleton of policy written in XACML format

In the implemented framework, all types of access control policies have been expressed in XACML but to incorporate authentication, privacy and trust based access control information, we are making use of its extension capabilities. To express privacy policies, "<purpose>" element has been introduced that will specify the purpose for which the service/resource can be accessed. Another addition that has been made is the inclusion of "<context>" element that will specify the context associated with a particular access request. Any of the access control policy (authentication, privacy, trust and policy)

can make use of this element if required. To express rest of the policies, we do not require any addition to standard XACML specification available. Figure 4.5 shows the placement of "<purpose>" and "<context>" elements in standard XACML policy.

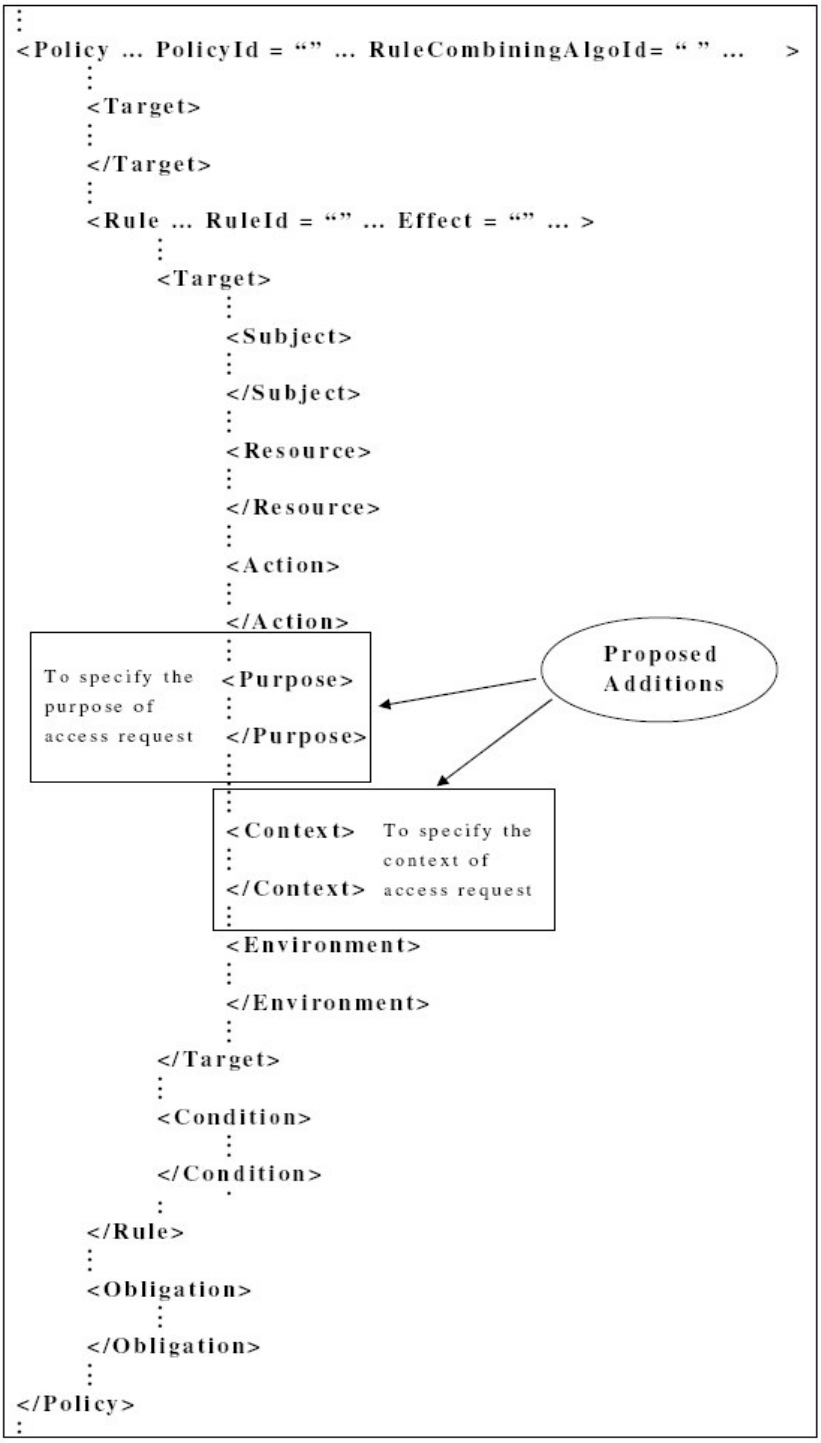

Figure 4.5: Skeleton of policy representing purpose and context attributes in XACML format

Figure 4.6 shows the skeleton of service policy file, which we are using in the security policy framework, to represent authentication, privacy, trust and authorization related access control policies applicable to a service/resource. Note that this file is different from XACML policy file but refer to other XACML policies through "Id" tag of "Policy" element. This file does not represent the complete set of policies applicable to that service/resource. As policies vary from subjects to subjects, the target element of other policies in the database is checked for the applicability of those policies to an access request.

```
⋮
<ServicePolicy ... PolicyId = "" ... PolicyCombiningAlgo= " " ...>
      ⋮
      <AuthenticationPolicy>
            ⋮
            <Policy ... Id = "" ...>
                  ⋮
            </Policy>
            ⋮
      </AuthenticationPolicy>
      ⋮
      <PrivacyPolicy>
            ⋮
            <Policy ... Id = "" ...>
                  ⋮
            </Policy>
            ⋮
      </PrivacyPolicy>
      ⋮
      <TrustPolicy>
            ⋮
            <Policy ... Id = "" ...>
                  ⋮
            </Policy>
            ⋮
      </TrustPolicy>
      ⋮
      <AuthorizationPolicy>
            ⋮
            <Policy ... Id = "" ...>
                  ⋮
            </Policy>
            ⋮
      </AuthorizationPolicy>
      ⋮
</ServicePolicy>
⋮
```

Figure 4.6: Skeleton of policy file associated with a service/resource

The other policies that describe requirements of a service like the types of security credentials accepted by service *i.e.* whether the service accepts X.509 certificates / Kerberos Tickets / Custom Security Token / Anonymous Security Token / username: password pair or whether the service is hidden service / private service / accepts anonymous access or whether the service provide trust based access *etc.* have been expressed in simple XML. These policies are attached to service/resource through service description file *i.e.* WSDL using customized "PolicyReference" element so that these requirements can be found while discovering service itself.

This chapter is mainly concerned with the expression mechanisms of different types of policies and not with their evaluation and enforcement. Next chapter presents the security policy framework along with its different models. The framework can be used to address key security requirements related to authentication, privacy, trust, authorization and to provide policy based access to grid services/resources. The chapter also describes the evaluation and enforcement of access control policies discussed in this chapter.

Chapter 5

Security Policy Framework

This chapter presents the implemented security policy framework. The key security requirements and policies of grid systems have been categorized under four heads namely authentication, privacy, trust and authorization. The previous chapter has explained different types of access control policies and the mechanisms used for their specification. This chapter presents models for handling each of the security requirements and policies related to authentication, privacy, trust and authorization together with their integration. Sections 5.1 to 5.3 present the individual authentication, privacy and trust models. Section 5.4 presents the policy based authorization framework and also discusses the integration of these models.

5.1 Authentication Model

Authentication system can be defined as a system that takes care of authentication related issues like single sign-on, delegation, nonrepudiation, secure logging, management of multiple user credentials *etc.* It is a detailed description of all aspects of a system that relate to authentication. Following sections describe how these features have been incorporated in the framework.

5.1.1 Single Sign-on and Delegation

The main problems related to authentication are of single sign-on and delegation. Single sign-on and delegation features depend on the type of security credentials used for authentication. *e.g.* these features can be easily provided, if the subject uses X.509 certificates, but, can not be provided if the subject is using username: password for

authentication. These features have been implemented in the framework through X.509 proxy certificates. Subjects are issued normal certificates by certificate authorities. Certificates bind subject's DN (Distinguished Name) to its public and private key. Public key is written in the certificate whereas private key remains with the subject. In order for subjects to prove their identity they must have i) Certificate and ii) Private Key. Authentication process involves presenting the certificate and proving the possession of the private key. Private key is held by subjects and is not disclosed to anyone under any circumstances. Proxy certificates differ from normal X.509 certificates in the sense that these are created by subjects and sent to other subjects/services so that they can act on original subject's behalf. Figure 5.1 shows how proxy certificates are created and used by a proxy service to act on original subject's behalf.

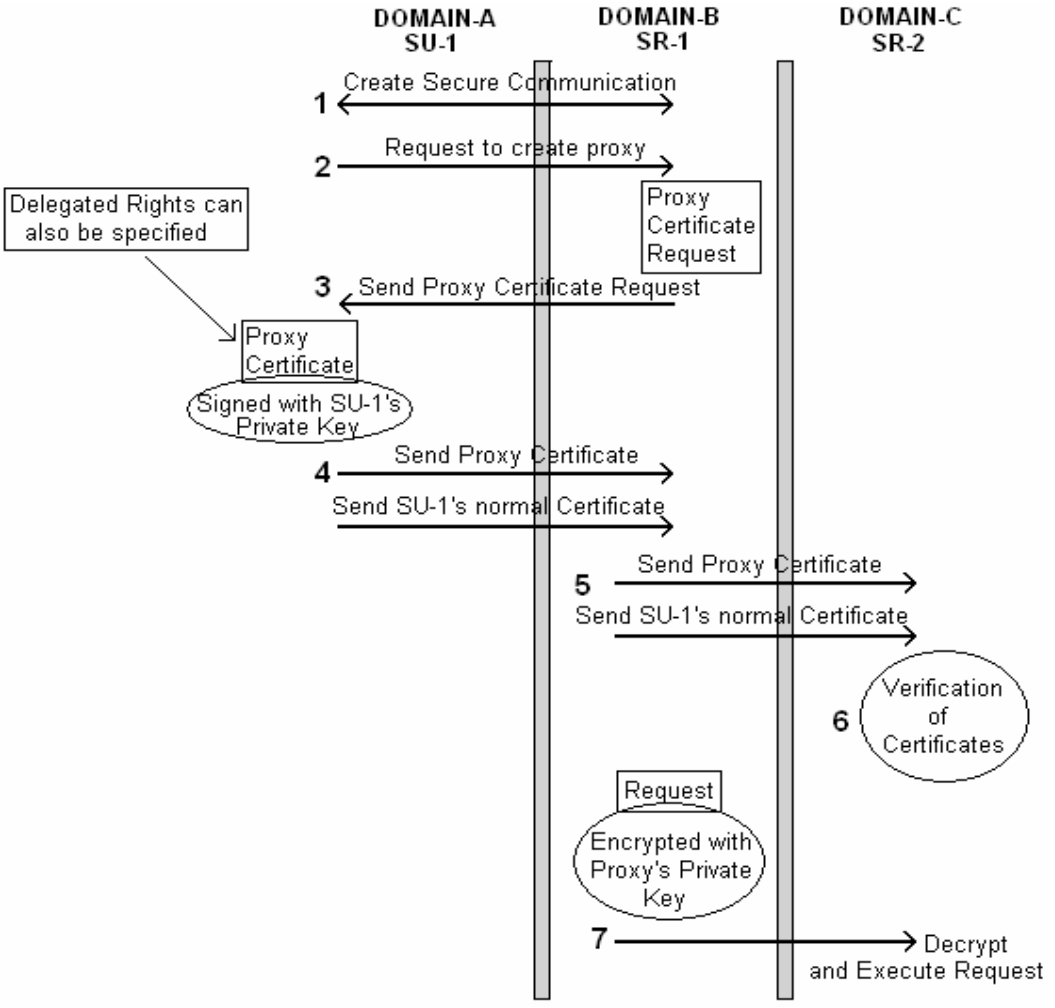

Figure 5.1: Proxy creation and action

Figure 5.1 shows SU-1 creating a proxy certificate and sending it to SR-1 so that SR-1 can act on its behalf. SR-1 can use this certificate to get authentication at other site on SU-1's behalf. Table 5.1 briefly explains the sequence of steps shown in Figure 5.1.

Step 1: A secure communication path is created between Subject SU-1 and the Service SR-1. Here SU-1 wants SR-1 to act on his behalf.
Step 2: SU-1 sends a request to create "proxy certificate request" to SR-1.
Step 3: A new key pair consisting of a public and private key is generated for use in proxy certificate. Using this public key, SR-1 creates a "proxy certificate request" and sends it to SU-1.
Step 4: SU-1 signs this "proxy certificate request" using his private key to create a proxy certificate. SU-1 then sends this proxy certificate to SR-1 along with his normal certificate.
Step 5: SR-1 sends SU-1's normal certificate and proxy certificate to other Service SR-2 to authenticate himself to SR-2 on SU-1's behalf. SR-1 is now acting as SU-1's proxy.
Step 6: SR-2 can verify the proxy certificate as it has SU-1's normal certificate, SU-1's public key (available through SU-1's normal certificate) and proxy certificate (signed using SU-1's private key).
Step 7: SR-1 can now send request to SR-2 on SU-1's behalf by encrypting that request with proxy certificate's private key. SR-2 can easily decrypt it as it has the proxy certificate's public key.

Table 5.1: Sequence of steps for proxy certificate creation and usage

In the above interactions, SU-1 uses his private key only once (while creating proxy certificate) and SR-1 is able to authenticate himself to SR-2 on SU-1's behalf. Delegation is also possible through proxy certificates. The delegated rights are expressed in XML and this information is attached to proxy certificates through PCI (Policy Certificate Information) field [63]. Subjects explicitly specify the subset of their total rights (which is to be delegated) in XML and this information is embedded in proxy certificates through PCI. Thus it is possible to support different delegation levels (*e.g.* impersonation, restricted delegation *etc.*) using proxy certificates. Step 3 in Figure 5.1 shows SU-1 specifying delegated rights in proxy certificate.

The authentication framework also supports trust based delegation in which rights are delegated only if trust with the target service/subject is greater than the threshold value. To calculate trust value, the pseudo codes described in Table 5.7, Table 5.8 and Table 5.9 are used. These are explained in Section 5.3.

5.1.2 Nonrepudiation, Confidentiality, Integrity and Secure Communication

Proxy certificates enable us to implement non repudiation also. As the subjects are digitally signing the certificates, they cannot deny from having sent the access request, as signature is possible only through their private key. Other security requirements like confidentiality and integrity of messages are guaranteed through encryption and signature techniques. For these, we are using XML-Encryption and XML-Signature features provided by .NET framework. For secure communication, we are making use of WS-SecureConversation specification implemented by WSE 3.0 toolkit under .NET environment. As already a lot of work has been done in the areas of confidentiality, integrity and secure communication, we are making use of existing technologies for these features. In addition to creation of proxy certificates, the implementation also supports the creation of custom security tokens and anonymous security tokens. Their purpose and use is explained in Section 5.2 while discussing privacy model.

5.1.3 Credential Management

As already mentioned, in grid, security requirements of different services can be different. *e.g.* In Domain A, Service-1 may require username/password, Service-2 may require Kerberos tickets and Service-3 may require X.509 certificates to authenticate subjects and so on. This is because services in a domain are provided by different service providers who generally belong to different physical organizations and they are free to implement any authentication mechanism of their choice to protect their services/resources. It is certainly neither possible nor a good idea to force different service providers to use same kind of security credentials to authenticate subjects. Though it may have advantages in

implementing a more robust authentication system but then the very purpose of integrating heterogeneous environments is lost.

Given this scenario, the main problem for the users is to satisfy the security requirements of different services. There are two approaches to the solution of this problem. First, we can provide a service to convert security credentials of one type to security credentials of other type. This approach looks fine as a first attempt but fails when we come across two incompatible credentials. *e.g.* it may be possible to convert a Kerberos ticket to equivalent X.509 certificate but it is certainly not possible to convert username: password to X.509 certificate or Kerberos Ticket. Therefore, we are following a different approach here. We want the subject to have different credentials to access different services. If a subject has to access different services from different domains then he must have all those types of security credentials as demanded by those services. One of these can be provided to the service according to the requirements of that service. Clearly, the second approach seems to be more suitable. But with this requirement, subjects will end up having many security credentials. Forcing users to manage credentials manually for each service is a tedious, error-prone and insecure task. The main problem here is the management of multiple user credentials.

As a solution to these problems, we propose grid domains to have a service to store, supply and maintain multiple security credentials for all of their subjects. The service will be responsible for managing subjects' multiple credentials and will be provided by the domain to which the subject belongs. In the implemented model, this service has been called Credential Manager Service (CMS). Subjects store all of their security credentials (X.509 certificates, Kerberos tickets, username: password pair *etc.*) in CMS.

To access a service, subject first discovers the security requirements of that service. Then subject checks the required security credentials in CMS. If the security credentials are present in CMS, then subject fetches those credentials from CMS by authenticating himself to CMS. If the required credentials are not in CMS, then subjects obtain them from the relevant certificate authority or from any other third party. The obtained credentials are then used by the subject to access the service. The new credentials are also stored in the CMS for future use. Figure 5.2 shows the usage of CMS where a Subject SU first discovers the security credentials required by Service SR using a discovery service. Subject then authenticates himself to CMS using username: password and fetches X.509 certificate. This certificate is then used by the Subject SU to access Service SR.

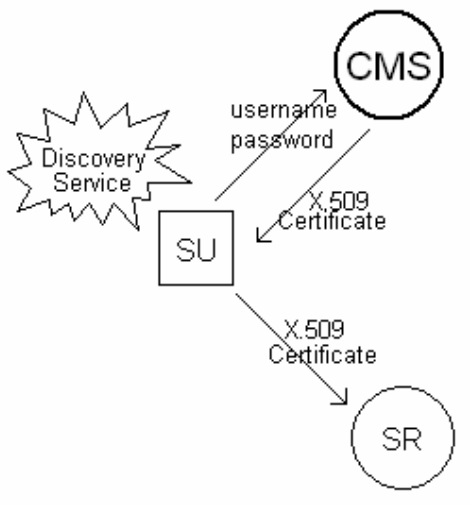

Figure 5.2: Credential Retrieval using Credential Manager Service

Following steps represent how a subject retrieves credentials using CMS.

Step 1: Subject discovers the security requirements of the service.
Step 2: Subject authenticates himself to CMS using the chosen credentials/mechanism.
Step 3: If authentication succeeds, subject checks CMS for the required credentials and performs the following:
a) If credentials are present, subject fetches them and goes to step 4.
b) If credentials are not present then subject obtains them from appropriate certificate authority.
c) Subject stores the credentials obtained from certificate authority in CMS for future use.
Step 4: Subject accesses the service using fetched credentials.

Table 5.2: Sequence of steps representing how a subject retrieves credentials using CMS

Figure 5.3 shows another example usage of CMS representing single sign-on and delegation also. A subject accesses CMS through username: password and CMS in turn, returns either X.509 Certificate/Kerberos Ticket as demanded by the subject. These credentials are passed to subject as SOAP messages. This interaction makes use of WS-Security specification. Subject then provides these credentials to Service SR-1 to access it. SU and SR-1 can also create a proxy certificate with/without right delegation to enable

SR-1 to access SR-2 on SU's behalf. Figure 5.3 also shows SR-1 passing proxy credentials to SR-2 to achieve single sign-on and delegation requirements.

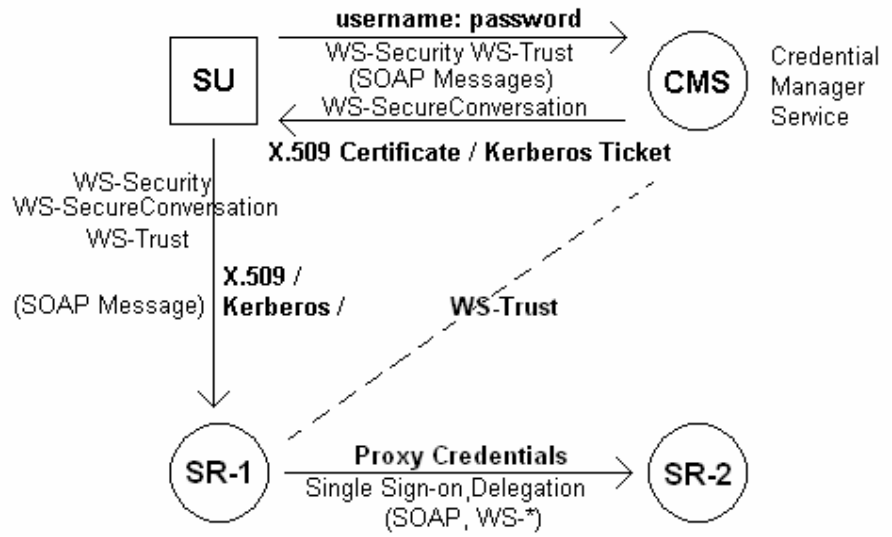

Figure 5.3: Usage of Credential Manager Service

Proxy credentials can also be stored in CMS. CMS can be used to store subject's private key also but in that case access to CMS will be through X.509 certificate and private key file is protected by password known only to the subject who owns the private key. If private key file is not stored in CMS then access to CMS can be provided through username: password also.

5.1.4 Services defined, implemented and exposed by Authentication Model

Authentication model has been implemented as a collection of authentication related services. Some of the important services defined, implemented and exposed by the authentication model are listed below:

Service Name	**Purpose**
upUpload( ):	To upload username: password pair.
x509Upload( ):	To upload X.509 certificate.
kerberosUpload( ):	To upload Kerberos security credential.

customUpload( ):	To upload custom security credential.
anonymousUpload( ):	To upload anonymous security credential.
proxyUpload( ):	To upload proxy certificate.
createProxy( ):	To create a proxy certificate.
searchCredential( ):	To search a particular credential from repository.
getup( ):	To retrieve username: password pair from repository.
getX509( ):	To retrieve X.509 certificate from repository.
getKerberos( ):	To retrieve Kerberos security credential from repository.
getCustom( ):	To retrieve custom security token from repository.
getAnonymous( ):	To retrieve anonymous security token from repository.
getProxy( ):	To retrieve proxy certificate from repository.
getAuthenticationPolicy( )	To retrieve authentication policy associated with a service.
setAuthenticationPolicy( )	To set authentication policy associated with a service.
removeUP( ):	To remove username: password pair from repository.
removeX509( ):	To remove X.509 certificate from repository.
removeKerberos( ):	To remove Kerberos security credential from repository.
removeCustom( ):	To remove custom security credential from repository.
removeAnonymous( ):	To remove anonymous security credential from repository.
removeProxy( ):	To remove proxy certificate from repository.
verifyUP( ):	To verify username: password pair.
verifyX509( ):	To verify X.509 certificate.
verifyKerberos( ):	To verify Kerberos security credential.
verifyCustom( ):	To verify custom security credential.
verifyAnonymous( ):	To verify anonymous security credential.
verifyProxy( ):	To verify proxy certificate.

Table 5.3: Services defined, implemented and exposed by authentication model

All these services have been implemented as web services in .NET environment. Table 5.3 lists only the service name and its purpose. The return type and arguments are not mentioned as most of the services exposed are overloaded. *e.g.* upUpload() service can accept two string values representing username and password for upload, as well as a

file (can be xml file also) containing username and password for upload. Similarly using searchCredential( ) service, subject can search for X.509 Certificates/Kerberos Tickets/Anonymous Security Credentials or any other type of credential depending on the arguments supplied to searchCredential( ) service. Moreover the services listed here are only the subset of the total services exposed by the authentication model. These services make use of other low level and high level services to achieve their tasks.

5.2 Privacy Model

A Privacy Model can be defined as a system that allows service requesters and service providers to state, evaluate and enforce their privacy requirements. It is a detailed description of all aspects of a system that relate to privacy. A privacy based authorization system grants specific type of access to specific requesters based on their authentication, what services/resources they are accessing, current state of the system and their conformance to established privacy policies. Privacy Model should be integrated with the authorization framework to provide privacy based access to services. Following section presents the details of the Privacy Model.

5.2.1 Privacy Infrastructure

We have discussed in Chapter 2 that privacy is becoming an important concern in grid and web environments where a large number of users provide their private/public information to service providers to gain access to their services/resources. The privacy model implemented by service providers must have the ability whereby users have control to check the misuse of their data as well as control over how their information can be collected, stored, used and shared. The privacy model should also allow service providers to express, evaluate and enforce privacy policies on their services/resources to protect them. A Privacy Model should address a wider notion of privacy and must be able to enforce privacy requirements of both the service providers and service requesters. It should also be integrated with the authorization framework to provide privacy based access to services/resources. To achieve this, we propose grid domains to implement privacy infrastructure as shown in Figure 5.4.

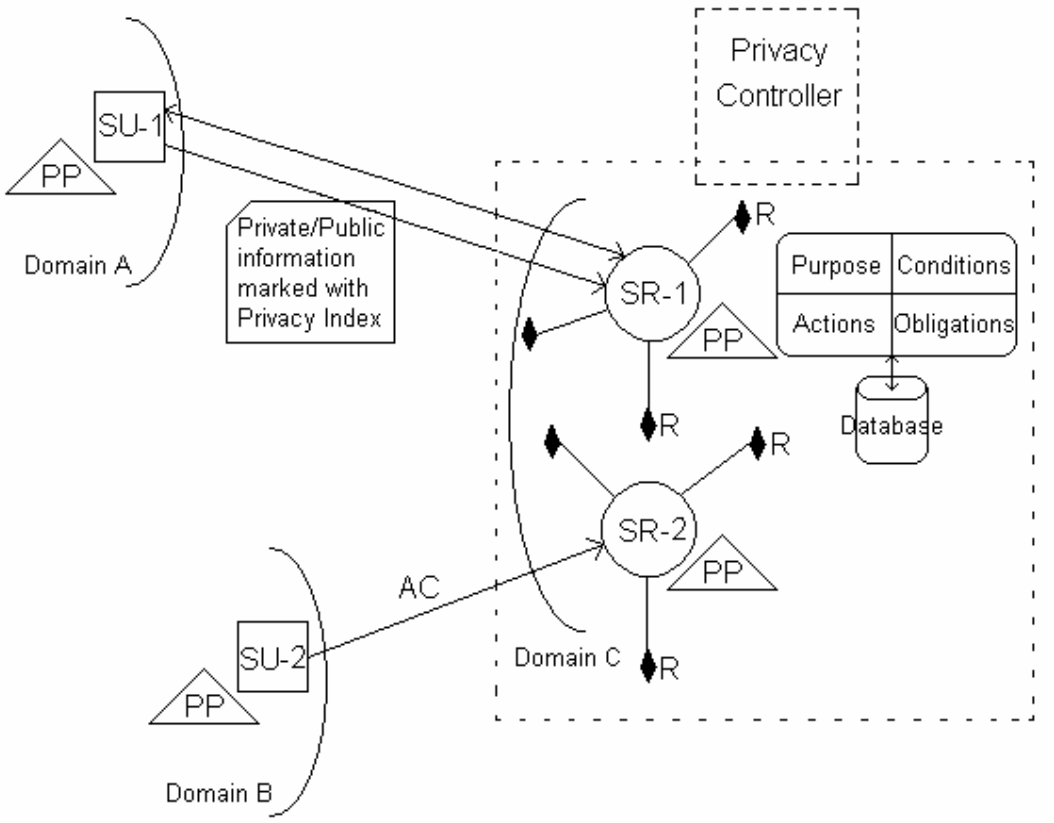

Figure 5.4: Schematic representing grid environment consisting of three domains showing how privacy requirements among subjects and services of different domains are handled.

Figure 5.4 represent a Grid Environment consisting of three Domains: Domain A, Domain B and Domain C. To understand how privacy requirements among service requesters and providers are handled, consider that SR-1 is a Service that asks for SU-1's private information. Further consider that this information is required by SR-1 to provide some other services to SU-1. Now before sending his private information to SR-1, SU-1 negotiates with SR-1 about his privacy requirements and establishes a privacy relationship. During negotiation, both agree on the information to be exchanged, the purpose for which information is required / can be used, the conditions under which information can be used, the duration for which information can be used and the parties with which information can be shared *etc.* SU-1 also marks his private information / data with PI (Privacy Index) while sending it to SR-1. All these privacy attributes *i.e.* private information, privacy index, purpose, conditions, duration, actions and obligations *etc.* are stored in database in the form of privacy policies expressed in simple XML and XACML. This interaction establishes a privacy relationship between SU-1 of Domain A and SR-1

of Domain C. In Figure 5.4, the direct association of Privacy Policies (PPs) with subjects and services is shown but actually these policies are associated with services/resources through Service Policy (SrP) because PP is a part of SrP.

A Privacy relationship has been represented as PR = (SU_a, DO_a, [R], [PU], [ACT], [CO], [OB], [SU_b], DO_b) *i.e.* Subject SU_a of Domain DO_a authorizes the set of Subjects [SU_b] of Domain DO_b to perform any of the actions specified in Action list [ACT] on set of Resources [R] for any of the purpose specified in Purposes set [PU] only if conditions specified in Conditions set [CO] are satisfied and obligations [OB] are performed. Notation "[x]" indicates the set of objects of type x. To specify more fine grained privacy relationship, PR can also be represented as (SU_a, DO_a, R, PU, ACT, CO, OB, SU_b, DO_b) i.e Subject SU_a of Domain DO_a authorizes Subject SU_b of Domain DO_b to perform Action ACT on Resource R for Purposes PU only if condition CO is satisfied and obligation OB is performed. Privacy relationships are used by authorization handler to provide privacy based access to services/resources.

The privacy relationships are kept by Domain C in such a way that later SU-1 can control and check their misuse. This is done through Privacy Controller (PC). PC is the way through which service requesters can be sure that their private information cannot be used inadequately by service providers. Whenever a privacy relationship is established between a subject and a service provider, a copy of this privacy relationship is sent to PC so that later PC can check (against this privacy relationship) whether service providers are respecting the privacy relationship established with the original subject or not.

Now consider that SU-1's private information is required by subject of some other domain, say SU-2 of Domain B. Further consider that SR-2 is a service of Domain C that provides access to SU-1's private information. Now accesses of SR-2 to SU-2 is provided only if this information is required by SU-2 to perform his task and the purpose of his task matches with the purpose for which this information was sent by SU-1. If the purpose for which SU-2 requires SU-1's private information does not match with the purpose for which SU-1 provided this information, then access is denied. To implement this functionality, the pseudo code defined in Table 5.4 has been used. A notification regarding the use of SU-1's private information is sent to SU-1, if he has asked for it. This requirement has been implemented through the concept of obligations. The violation of any established privacy relationship is checked by Privacy Controller (PC).

Following are some of the examples illustrating how privacy relationships are represented using the notation described above.

Policy 1: I authorize marketing people of organization to use my telephone number to inform me about new offers only: PR = (I, Home, telephone-no, inform-new-offer, read, null, null, marketing-people, organization)

Policy 2: Banks authorize only owners of accounts to change their personal information any time if their age is greater than 18: PR = (Manager, Bank, personal-information, null, read/write, age>18, null, owner-account, Home)

Policy 3: ABC-Travel agency will share personal information of users with government only if ordered by court and users will be notified: PR = (Manager, ABC-Travel, personal-info-users, null, read, order-by-court, notify, officials, Government)

The privacy relationships lead to privacy policies on services. The privacy policies that describe the privacy requirements of a service have been expressed in XML and privacy based access control policies have been expressed in XACML. The skeleton of these files has already been presented in Chapter 4. Section 5.2.3 describe how privacy based access to grid services/resources is provided. Privacy Model's integration with the authorization framework is explained in Section 5.4.

5.2.2 Hidden/Secret Services and Anonymous Access

Though PC plays an important role in handling service requesters' privacy requirements, it cannot be used by service providers to address their privacy requirements. To handle service provider's privacy requirements, TTP is used.

To understand how TTP has been used to address service provider's privacy requirements, consider that a service provider wants to expose a service only to a selected set of subjects and also wants to hide the security requirements associated with that service (*i.e.* the security credentials/attributes required to access the Service). Such services are called hidden/secret services. In order to provide this feature, we propose service provider to make use of TTP. Service providers will share authentication information of subjects and security requirements of services with TTP. To access a hidden/secret service, subjects pass their normal security credentials and attributes to TTP. TTP authenticates subjects (as service providers share authentication information with TTP) and if authenticated, based on the credentials/attributes provided by subjects,

TTP generates a custom security token. In custom security token, only those credentials and attributes are included which are necessary to access the service. This security token is then sent to subjects. Only this security token can be used by subjects to see and access hidden/secret services. If the subject is not authenticated or not authorized to access hidden/secret services then no custom security token is generated. Based on the information stored in custom security token, the service provider exposes some/all of the hidden/secret services that the subject is authorized to access. As subject has no knowledge of custom security token, he cannot know the actual security requirements of hidden/secret services and the actual credentials/attributes that cause access to these services. The custom security token has only limited life time and includes only those minimum credentials and attributes of the subject that are necessary to access the hidden/secret services. This mechanism provides two important privacy features to service providers:

i) Hidden/Secret services and their security requirements are not exposed to everyone.

ii) Even the authorized subjects don't know the exact security requirements of hidden/secret services. As subjects are given custom security token, they can't deduce which subset of their credentials/attributes is embedded in the custom security token and is causing access to services.

TTP also enables anonymous access on services. As service providers share authentication information of subjects with TTP, TTP can be used to authenticate subjects and generate anonymous security tokens. Anonymous security tokens assert about authorization information of subjects but remove their identity. These anonymous security tokens are used by subjects to get anonymous access to services. As anonymous security token has no information regarding the identity of subjects but only about their authorization information, service providers can't know exactly which subject is accessing the service.

Figure 5.5 shows how anonymous access to a service is provided. Here Subject SU wants to access Service SR without disclosing his identity to service provider who provides this service in Domain B. For this, SU requests TTP to generate an anonymous security token based on his identity and access rights. As service provider shares authentication and authorization information with TTP, TTP verifies the identity of SU and generates an anonymous security token. This token has limited lifetime and asserts

about authorization of SU. Thus this token can be used by SU to access service SR without disclosing his identity to service provider.

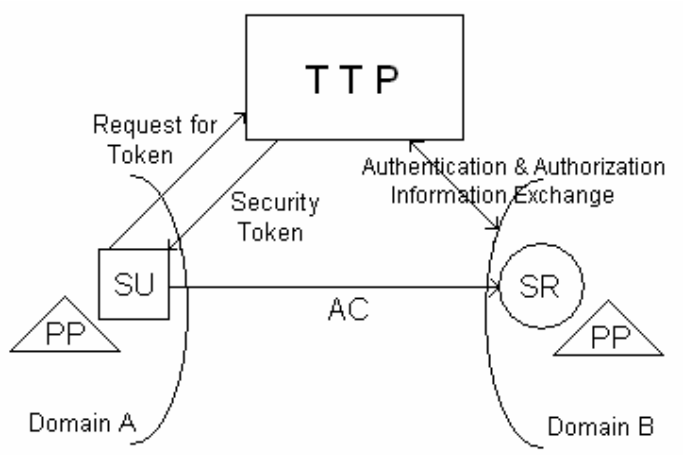

Figure 5.5: Schematic showing how anonymous access to a Service is provided

The same setup is used to access hidden/secret services also. If SU wants to access a secret service provided by Domain B then he requests a custom security token from TTP by presenting him his normal security credential. TTP verifies the identity of SU, fetches SU's attributes from Domain B, generates a custom security token and sends it to SU. Now SU can present this custom security token to Domain B to see and access all hidden/secret services that he is authorized to access.

5.2.3 Privacy based Access

The Privacy Model also supports privacy based access (or more precisely purpose based access). The established privacy relationships can be used by authorization framework to provide privacy based access to services. The following sequences of steps have been used in the model to provide privacy based access to a service/resource when a subject requests it for some purpose.

Step 1: Extract security credentials of the Subject from request and verify the identity. If Subject authentication fails then quit otherwise go to Step 2.
Step 2: Determine whether Subject conforms to Privacy Policy (PP) associated with that

Service/Resource. For this, perform the following:

i) Get the list of authorized Subjects ([SU]) that can access the requested Service/Resource.

ii) If Subject's identity does not appear in the list of authorized Subjects then quit otherwise do the following.

iii) Obtain the purpose set [PU] associated with the requested Service/Resource for which it can be accessed.

iv) Extract the purpose of access from access request.

v) If the purpose for which Subject wants to access the Service/Resource is not listed in the set of purposes ([PU]) then quit otherwise perform the following.

vi) If the access action specified in the Subject's access request is not listed in the set of actions allowed ([ACT]) on that Service/Resource for that particular purpose then quit otherwise perform the following.

vii) Obtain privacy statements and conditions associated with that Service/Resource from the policy database and, the Subject and Resource attributes from the attribute database.

viii) Check whether the conditions specified in [CO] set are satisfied. If not then deny access otherwise proceed next.

ix) Check whether the obligations specified in the [OB] set can be performed. If not, then quit otherwise perform Step 3.

Step 3: Provide access of Service/Resource to Subject.

Step 4: Perform obligations specified in the [OB] set.

Step 5: Log access actions in log tables to address auditing and accounting requirements.

Table 5.4: Sequence of steps for privacy based access to grid services/resources

The above sequence of steps guarantee that if a service is exposing subject's private information then that information can be accessed by other subjects/services only if they are authorized and the purpose for which they are accessing the information matches with the purpose for which the information was actually sent by the subject to the service provider. These steps are integrated with overall authorization framework to provide privacy based access to services/resources. As access to resources depend on many other factor also (besides privacy), authorization framework make use of other relevant components to arrive at final authorization result.

Privacy Controller can be used to make sure for subjects that service providers are providing privacy based access to private information according to the privacy relationships that they have established initially. Thus subjects can be sure that their private information can not be misused / accessed inadequately from service provider's database.

5.2.4 Services defined, implemented and exposed by Privacy Model

Privacy model has been implemented as a collection of privacy related services exposed as web services. Some of the services defined, exposed and used in the implementation are:

Service Name	**Purpose**
getAuthorizedSubjects( )	To get a list of authorized subjects associated with a service for a particular purpose for a particular action.
getPurpose( )	To get purpose associated with a service for a particular action.
getAllowedAction( )	To get allowed actions on a service for a particular purpose.
getPrivacyPolicy( )	To get privacy policy associated with a service.
getPrivateServices( )	To get a list of services that require private data.
getAnonymousServices( )	To get a list of services that can be anonymously accessed.
getSecretServices( )	To get a list of services that are hidden.
getPrivacyIndex( )	To get privacy index associated with a resource/service.
setAuthorizedSubjecst( )	To set authorized subjects for a particular service for a particular purpose and action.
setPurpose( )	To set purpose associated with a service for a particular action.
setAllowedActions( )	To set allowed actions on a service for a particular

	purpose.
setPrivacyPolicy( )	To set privacy policy associated with a service.
setPrivacyIndex( )	To set privacy index associated with a service /resource.
isPrivate( )	To find whether a service requires private data.
isAnonymous( )	To find whether a service supports anonymous access.
setPrivacyType( )	To set the privacy type of a service (hidden / anonymous / private *etc*.)
removeAuthorizedSubjects( )	To remove subjects associate with a service for a particular action or purpose.
removePurpose( )	To remove purpose associated with a service for a particular action.
removeAllowedActions( )	To remove allowed actions associated with a service for a particular purpose.
removePrivacyPolicy( )	To remove privacy policy associated with a service.

Table 5.5: Services defined, implemented and exposed by privacy model

These services have been implemented in .NET environment as web services. Most of the services listed in Table 5.5 are overloaded and represent a subset of the total services exposed by the privacy model.

5.3 Trust Model

A Trust Model can be defined as a system that allows service requesters and service providers to assess trustworthiness of each other as well as state, evaluate and enforce trust relationships among them. It is a detailed description of all aspects of a system that relate to trust. A trust based authorization system grants specific type of access to specific requesters based on their authentication, trust status, what services/resources they are accessing, current state of the system and their conformance to established trust and other security policies. Trust Model can also be integrated with the authorization framework to

provide trust based access to services. Following sections present the details of the implemented Trust Model.

5.3.1 Trust Relationships

Determining trust relationship is essential not only while accessing resources/services but also while enabling delegation. A subject may decide to delegate rights to target service/subject only if trust with that service/subject is greater than a threshold value. A subject may also decide to send his/her private data/information to target service/service provider/domain based on the value of trust it has with the target service/service provider/domain. Trust can take a complex form in distributed environment. The trust status of a service/resource from a different domain is hard to determine. Subjects generally have no idea whether the service is compromised or is malicious [103]. There may occur different trust relationships among members of different domains. *e.g.* Subjects of Domain A trust Domain B and not Domain C, or, Subjects trust Domain B for Service 1 but not for Service 2, *etc.* The main problem is how to establish and manage trust relationships between subjects and services of different domains and how to determine the trustworthiness of target service/service provider/domain. Domains need a trust enabled security infrastructure to handle trust related requirements and to provide trust based access to services/resources. Establishment of trust may be a one time activity per session or it may be dynamic [43] (requiring the establishment of trust for each request). The trust of an entity with other entity is not a fixed value but can change dynamically depending on the past and present behavior of the entity and context in the environment.

Trust should be established from the viewpoint of both the parties (Service Requesters and Service Providers). Requester's trust with the service provider may be different from the service provider's trust with the requester. This situation is modeled using two different types of trust: code trust and execution trust, as defined in [103]. Execution trust exists from subject's side to service provider's side that service provider will correctly and faithfully allocate resources for the efficient execution of a job with respect to established trust and other security policies. Code trust exist from service provider's side to subject's side that subject will generate a legitimate request consisting of virus free code and will not produce malicious results and does not temper other

results/information/code present at service provider's end. In addition to execution and code trust, following terminology regarding trust has also been used in the security policy framework:

Direct Trust: is the trust that a subject holds on a service /service provider/ domain without any intermediate service/service provider/domain or other entity.

Full Trust: A subject is said to have full trust on a service provider/domain if it trust on all the services provided by that service provider/domain.

Partial Trust: A subject is said to have partial trust on a service provider/domain if it trusts on some of the services provided by that service provider/domain.

Recommended Trust: is the trust of one entity on second entity that is recommended by other entities.

Privacy Trust: is the trust of an entity on the privacy features provided by other entity.

Authentication Trust: is the trust of an entity on the authenticity of an identity certificate signed by a certificate authority.

We propose trust relationship to be represented as: TR = (SU_1, DO_1, TT, SR_1, DO_2, c, t, tv) *i.e.* Subject SU_1 of Domain DO_1 trust Service SR_1 of Domain DO_2 with trust value tv with respect to trust type TT at time t for context c. TT represent the type of trust which can be Authentication Trust, Privacy Trust, Authorization Trust, Direct Trust, Recommendation Trust *etc.* We also propose Trust Model to be a system represented as TS = (GES, TR, OP), where GES refers to different entities in a grid environment among which trust relationship exist. These entities can be subjects, services, service providers or domains *etc.* TR is the type of trust (as discussed above) that can exist among these entities and OP is the set of operators used to determine trustworthiness of one entity on another. Next section describes the approach used in the implemented model to determine the trustworthiness of target entity.

5.3.2 Trust Evaluation

If a subject from a particular domain wants to access a service from another domain, then trust value *tv* can be calculated as:

$$tv = \alpha * dt + \beta * rt \qquad \text{(1)}$$

where *dt* is Direct Trust and *rt* is Recommended Trust. α and β are the weights assigned to direct and recommended trust. In general, direct trust is assigned more weightage over recommended trust. α and β are chosen such that $\alpha + \beta = 1$. Let *s* and *f* denote the success and failure evidences experienced by the subject about the service. Then direct trust can be calculated as:

$$dt = s / (s + f) \quad \text{.....} \quad (2)$$

and recommended trust can be calculated as:

*rt = Average of (subject's trust on R1 * R1's trust on service,*
*subject's trust on R2 * R2's trust on service,*
.....
.....
*subject's trust on Rn * Rn's trust on service)* ..... (3)

where *R1, R2, …, Rn* are recommenders of the subject. Recommenders are trusted by subjects to make recommendations about trustworthiness of other services. We are making use of these equations to determine the trustworthiness of one entity on other entity. In case of full trust, *f* can be set to 0 and in case of partial trust, *s* and *f* take different values depending on the level of trust. In case of no trust, *s* can be set to 0. A history of these values is maintained by every domain. To calculate *s* and *f*, last *n* interactions are considered where *n* can be any number depending on the requirements. A large value of *n* models accurate behavior whereas a small value of *n* models current behavior of the entity. To update trust, changes are made to *s* and *f*. Based on the values of *dt* and *rt*, trustworthiness of an entity can be categorized into different levels as shown in Table 5.6.

Level	***dt***	***rt***	***tv* (with α=β=0.5)**	**Meaning**
L1	>0.5	>0.5	>0.5 and <=1	Trust (White Zone)
L2	>=0.5	<=0.5	<=0.75	Can be trusted with some risk (Gray Zone)
L3	<=0.5	>=0.5	<=0.75	Can be trusted but greater risk (Gray Zone)
L4	<0.5	<0.5	<0.50	Don't Trust (Black Zone)

Table 5.6: Levels of trust with α = 0.5 and β = 0.5

Table 5.6 shows four levels of trust ranging from reasonable trust (White Zone) to no trust (Black Zone) depending upon the *dt* and *rt* values. *dt* and *rt* can take a maximum value of 1. *tv* can also take a maximum value of 1. The more the value of *tv* is, the higher is the trust. A *dt* and *rt* value greater than 0.5 means that both the subject and the recommender have more success evidences than failure. This represents a White Zone and target can be trusted with a satisfactory level of trust. If a subject experiences more success evidences and recommender experiences failure evidences, or vice versa, then the trustworthiness of target is doubtful. This represents Gray Zone. At this point, a decision should be made whether subject wants to give more weightage to recommender trust (*rt*) or his direct trust (*dt*) on service. In case where $\alpha = \beta = 0.5$ *i.e.* both, the direct and recommender trust are given equal weightage, these two levels are the same. Generally, *dt* value is given more weightage over *rt* value. So if a subject experiences more success evidences and recommender experiences failure evidences, then the trustworthiness of target is less doubtful compared to subject being experiencing failure evidences and recommender experiencing more success evidences. If the subject and recommender, both have more failure evidences than success evidences, then the target is definitely not to be trusted. This represents a Black Zone. Two important data structures that have been maintained for the calculation of trust value (*tv*) are:

i) Trust Table

ii) Recommender Table.

Each Domain maintains a copy of these tables. Following is the basic layout of these tables:

Trust (sourceDomain, sourceServiceProvider, sourceSubject, targetDomain, targetServiceProvider, targetSubject, success, failure, threshold, context, time)

Recommender (recommenderDomain, recommenderServiceProvider, recommenderSubject, context, time)

Using above equations and data structures, trust of a subject on subjects/services of other domains can be calculated. For this, we are making use of the pseudo codes presented in Table 5.7, Table 5.8 and Table 5.9 that make use of these equations and data structures.

calculateTrust (S-DO, S-SP, S-S, T-DO, T-SP, T-S, c)

S-DO: Source Domain

S-SP: Source Service Provider

S-S: Source Subject

T-DO: Target Domain

T-SP: Target Service Provider

T-S: Target Subject

c: context

tv: trust value

dt: direct trust

rt: recommended trust

α: direct trust weightage

β: recommended trust weightage

i) Set tv = 0, dt = 0, rt = 0, α = 0.6, β=0.4.

ii) Set dt= calDirectTrust (S-DO, S-SP, S-S, T-DO, T-SP, T-S, c)

iii) Set rt = calRecTrust (S-DO, S-SP, S-S, T-DO, T-SP, T-S, c)

iv) Set tv= α * dt+ β * rt

v) Return tv

Table 5.7: Pseudo code to calculate trust value

calDirectTrust (S-DO, S-SP, S-S, T-DO, T-SP, T-S, c)

S-DO: Source Domain

S-SP: Source Service Provider

S-S: Source Subject

T-DO: Target Domain

T-SP: Target Service Provider

T-S: Target Subject

c: context

dt: direct trust

i) Set dt = 0.

ii) Select success, failure from Trust where sourceDomain=S-DO and sourceServiceProvider=S-SP and sourceSubject=S-S and targetDomain=T-DO and targetServiceProvider=T-SP and targetSubject=T-S and context=c

iii) Set dt=success / (success + failure)

iv) Return dt

Table 5.8: Pseudo code to calculate direct trust

calRecTrust(S-DO, S-SP, S-S, T-DO, T-SP, T-S, c)

S-DO: Source Domain

S-SP: Source Service Provider

S-S: Source Subject

T-DO: Target Domain

T-SP: Target Service Provider

T-S: target Subject

c: context

nr: number of recommenders

i: loop variable

str: source's trust on recommender

rtt: recommender's trust on target

CR-D: current recommender Domain

CR-SP: current recommender Service Provider

CR-S: current recommender Subject

rt: recommended trust

i) Set rt = 0

ii) Set nr = Select count (*) from Recommender

iii) for (i =1; i<nr; i++)

a) Set CR-D= Select recommenderDomain[i] from Recommender

b) Set CR-SP = Select recommenderServiceProvider[i] from Recommender

b) Set CR-S= Select recommenderSubject[i] from Recommender

c) Select success, failure from Trust where sourceDomain=S-DO and sourceServiceProvider=S-SP and sourceSubject=S-S and targetDomain=CR-D and targetServiceProvider=CR-SP and targetSubject=CR-S and context=c

d) Set str = success / (success + failure)

e) Set rtt = getDirTrust (CR-D, CR-SP, CR-S, T-DO, T-SP, T-S)

f) rt = rt + str*rtt

iv) Set rt = rt/nr

v) Return rt

Table 5.9: Pseudo code to calculate recommended trust

Table 5.7 represents pseudo code to calculate trust value. As shown, the direct trust, recommender trust and trust value are initially set to 0. Direct trust weightage (α) and recommender trust weightage (β) are set to 0.6 and 0.4 respectively as generally direct trust is given more weightage over recommender trust. Different values of α and β can also be taken to implement different weighting strategy. Then the direct trust of the source subject on the target service is calculated using the calDirectTrust( ) algorithm, which is shown in Table 5.8. To calculate direct trust, the success and failure evidences experienced by the source subject with target service are fetched from the Trust table and *dt* is calculated using Equation (2). This is shown in step iii of calDirectTrust( ). To calculate recommended trust, calRecTrust( ) algorithm is used which is shown in Table 5.9. After determining *dt* and *rt* values, trust value *tv* is calculated using Equation (1). This is shown in step iv of calculateTrust( ) algorithm. To calculate recommended trust, first, the trust of the source subject on recommender is calculated and then the recommender's trust on target is called from the recommender. This is shown in steps iii-d and iii-e of calRecTrust( ) algorithm. The process is repeated for all the recommenders in the Recommender table. After getting the recommendation from all the recommenders, average value of *rt* is calculated using Equation (3). This is shown in steps iii-f and iv of calRecTrust( ) algorithm.

Using these algorithms, we can set up trust enabled authorization framework that determine the trustworthiness of one entity on another and provide access to target service based on the trust value. A threshold value of *tv* can be set depending on the level of trust required. After authenticating subject, we can calculate *tv* value and check whether *tv* is greater than or less that threshold value. If $tv \geq$ threshold value then access to service/resource is provided otherwise access is denied. After each access, following updation are made to *s* and *f* values:

$s = s + 1$, if source is satisfied by the service/resource access provided by the target, otherwise no change.

$f = f + 1$, if source is not satisfied by the service/resource access provided by the target, otherwise no change.

5.3.3 Trust Manager Service

In order to establish and evaluate trust among different entities in a grid environment, we propose every domain to implement trust model integrated with the authorization framework. Besides handling trust related security requirements and issues, trust model will also provide trust based access to services/resources. Figure 5.6 shows a high level view of the trust model.

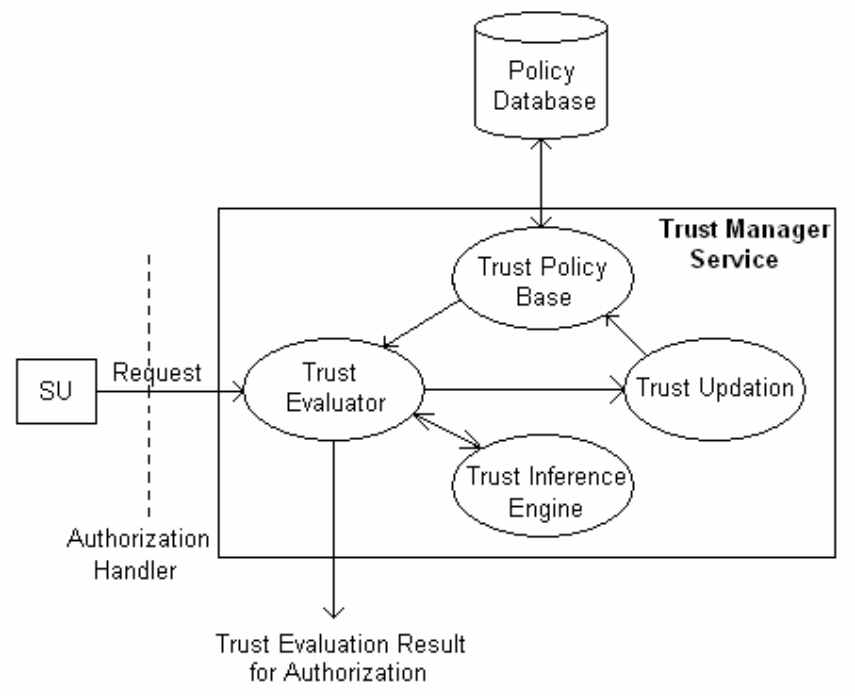

Figure 5.6: Trust Model architecture for trust based access

The Trust Model has been implemented as a Trust Manager Service. As shown in Figure 5.6, access request coming from subject, through authorization handler (explained in Section 5.4), is first intercepted by Trust Evaluator component. Trust Evaluator is responsible for evaluating trust based on established trust relationships and providing trust evaluation result to authorization handler. To evaluate trust, Trust Evaluator makes use of Trust Inference Engine. Trust Inference Engine calculates *tv* value and passes it to trust evaluator for use in preparing final result. Trust Evaluator is also responsible for updation of trust among entities. Trust is updated with every interaction that takes place between a subject and a service. Trust Policy Base fetches established trust relationships and policies from policy database and provides these to trust evaluator for evaluating trust

and preparing final result. Trust Evaluator then passes this result to authorization handler. The trust model is capable of capturing different types of trust and provides mechanism for trust evaluation, recommendations and updation of trust. Trust related access control policies have been expressed in XACML [41]. The format of XACML policy has already been discussed in Section 4.3. Trust model's integration with the authorization framework is described in Section 5.4.

5.3.4 Services defined, implemented and exposed by Trust Model

Trust model has been implemented as a collection of trust related services exposed as web services. Some of the services defined, exposed and used in the implementation are:

Service Name	**Purpose**
getTrustOnDomain( )	To get trust on a domain.
getTrustOnServiceProvider( )	To get trust on a particular service provider.
getTrustOnService( )	To get trust on a service.
getDirectTrust( )	To get direct trust on a service.
getRecommendedTrust( )	To get recommended trust on a service.
getRecommender( )	To get a list of recommenders for a service/service provider/domain.
getTrustPolicy( )	To get trust policy associated with a service.
getTrustThreshold( )	To get trust threshold associated with a service.
getTrustedDomains( )	To get a list of trusted domains.
getTrustedServiceProviders( )	To get a list of trusted service providers.
getTrustedServices( )	To get a list of trusted services.
getSuccessEvidences( )	To get success evidences experienced with a service for a particular context.
getFailureEvidences( )	To get failure evidences experienced with a service for a particular context.
getTrustContext( )	To get context associated with a trust relationship.
addRecommender( )	To add recommenders for a particular service/service provider/domain.

setTrustTheshold( )	To set trust threshold associated with a particular service.
setTrustPolicy( )	To set trust policy associated with a service.
removeRecommender( )	To remove recommenders for a particular service/service provider/domain.
removeTrustPolicy( )	To remove trust policy associated with a service.
updateTrustValues( )	To update trust values (success / failure evidences experienced) for particular service.

Table 5.10: Services defined, implemented and exposed by trust model

Like other services, these services have also been implemented as web services in .NET environment. Most of the services listed here are overloaded and represent a subset of the total services exposed by the trust model.

5.4 Policy based Authorization Model

As discussed in Chapter 2, authorization system can be defined as a system that grants specific type of access to specific requesters based on their authentication, what services/resources they are accessing, current state of the system and their conformation to established authentication, privacy, trust and other security policies. It is a detailed description of all aspects of a system dealing with access of services/resources by requesters. Authorization system should be scalable and must be able to enforce different types of security policies. Following sections present the details of the implemented framework.

5.4.1 Authorization System

In grids, the number of subjects and services/resources are very large. The services/resources also have different authentication, privacy, trust and authorization requirements. Subjects have different roles/privileges in different domains, which results in their different authorization status in different domains. If the number of requesters requesting the services of a target domain is very large then it will be very difficult for the

target domain to maintain access rights information of all of its requesters. To address this problem, we propose target domain to store access rights information of all of the users/requesters from a source domain as a whole (*i.e.* the access rights that target domain grants to source domain irrespective of a particular user/requester) and not of the individual users/requesters from the source domain. The source domains will themselves maintain access rights information of their respective subjects. This approach will make the authorization system more scalable. This approach has been implemented in the framework using Filter-In and Filter-Out components. Figure 5.7 illustrates this concept.

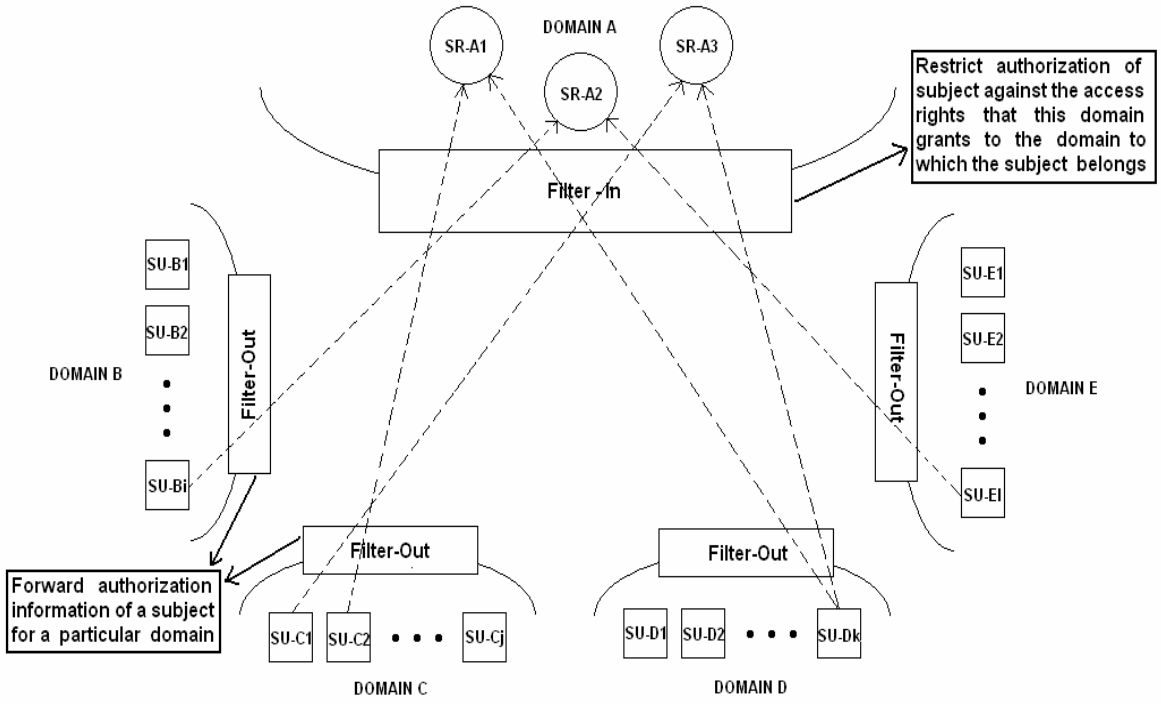

Figure 5.7: Schematic showing the use of Filter-In and Filter-Out components

As shown in Figure 5.7, there are five domains: A, B, C, D and E. The domains B, C, D and E have i, j, k and l number of subjects respectively. These subjects access services provided by domain A. Each of these subjects has different roles/privileges in domain A. If the number of subjects in domain B, C, D and E is very large then it will be very difficult for domain A to maintain access rights information of all of these subjects from different domains. As a solution to this problem, we propose domain A to store access

rights information of all the subjects of a particular domain as a whole and not of individual subjects of individual domain. This means that domain A will maintain

i) access rights information that it grants to domain B irrespective of particular subjects of Domain B

ii) access rights information that it grants to domain C irrespective of particular subjects of Domain C

iii) access rights information that it grants to domain D irrespective of particular subjects of Domain D

iv) access rights information that it grants to domain E irrespective of particular subjects of Domain E

The exact access rights information of a subject for services of Domain-A will be maintained by the domain to which the subject belongs.

To achieve this functionality, we are making use of Filter components. An example usage of Filter components has been explained in Chapter 3. There are two types of Filters: Filter-In and Filter-Out. Through Filter-Out component, the subject leaves his source domain with access rights that his source domain grants to him. It attenuates/filters the access rights of a subject for a particular target domain. Through Filter-In component, the subject enters the target domain with access rights that the target domain grants to the source domain. In other words, the subject gets the intersection of the rights that his source domain grants to him and the rights that target domain grants to the source domain. In this particular example, Filter-Out components of Domains B, C, D and E will maintain access rights information of their respective subjects and Filter-In component of Domain A will maintain access rights information that Domain-A grants to Domain B, Domain C, Domain D and Domain E irrespective of their subjects. This makes the authorization system more scalable as a single domain is not maintaining access rights information of all the subjects instead subject wide access rights information is maintained by the source domain to which subject belongs and the domain wide access rights information is maintained by the target domain.

Filter-In and Filter-Out components have been implemented as services which make assertions about access rights information of subjects and get called when subjects issue access request on a service.

5.4.2 Policy Based Access System

As described in Chapter 3, the access control policies of a Domain, Service Provider and Service have been expressed in XACML. The powerful XACML policy language model allow us to specify much fine grained access control policies on services/resources using its different elements, which is not possible through other mechanisms. To represent context of a particular access request, the addition of "context" element in XACML policy language model is proposed. Figure 4.5 illustrates the placement of proposed "context" tag in XACML policy language model. In the implemented framework, XACML and its extension capabilities are allowing us to specify fine grained and context based access control policies on services/resources.

After determining the authorization information of subjects, their conformance to domain policy (DP), service provider policy (SPP), service policy (SrP) and other applicable policies is checked. If subject conforms to all these policies, then access is granted otherwise access is denied. Following sequence of steps represent a high level description of the process of providing access to a requested service/resource.

Step 1: Extract security credentials of subject from request and verify the identity. If subject authentication fails then quit otherwise go to Step 2.

Step 2: Determine authorization information of subject using Filter components. If subject is not authorized to access the service then quit otherwise go to Step 3.

Step 3: To determine whether subject conforms to all applicable security policies, extract the Domain Policy (DP), Service Provider Policy (SSP), Service Policy (SrP) and the policies for which target matches with access request attributes from the policy database, and perform the following for each of the extracted policies:

i) Call authentication policy handler to determine whether subject conforms to authentication policies. If not then quit otherwise proceed next.

ii) Call privacy policy handler to determine whether subject conforms to privacy policies. If not then quit otherwise proceed next.

iii) Call trust policy handler to determine whether subject conforms to trust policies. If not then quit otherwise proceed next.

iv) Determine whether subject conforms to other access control policies. If not then quit otherwise proceed next.
Step 4: Check whether the obligations applicable for this access can be performed. If not, then quit otherwise perform Step 5.
Step 5: Provide access to requested service/resource.
Step 6: Perform applicable obligations.
Step 7: Log access actions in log tables to address auditing and accounting requirements.

Table 5.11: Sequence of steps for policy based access to grid services/resources

To achieve the functionality described above, we are proposing and making use of the XACML based authorization model as shown in Figure 5.8. As DP, SPP and SrP can have authentication, privacy and trust related policies also, the authentication, privacy and trust handler have also been integrated in the framework to evaluate their respective policies. The authentication handler functionality has been implemented in PDP itself so it is not shown in Figure 5.8 but privacy and trust handlers are explicitly shown. The major components of the model are PEP, PDP and PIP. PEP (Policy Enforcement Point) performs access control by making decision requests and enforcing authorization decisions. PDP (Policy Decision Point) evaluates applicable policy and renders an authorization decision. PIP (Policy Information Point) acts as a source of attribute values.

The authenticated and filtered request of Subject SU is first intercepted by PEP. PEP constructs an authorization decision query and passes it to authorization handler. The result of this query determines whether access to service/resource is granted or denied. The authorization decision query has details about the identity of the subject, the service requested, the purpose of access *etc.* Authorization handler passes this information to PDP. PDP is responsible for evaluating policies. The Policies are retrieved by PDP from PRP (Policy Retrieval Point). If the policy information is not available at PRP, it may be retrieved from Policy Store. The Policy Store is capable of importing/exporting policies. The policies are written by administrator using PAP (Policy Administration Point).

PDP is proposed to be implemented as a combination of other PDPs (PDP-1, PDP-2 … PDP-n). These components (PDP-1, PDP-2, … PDP-n) will implement policy decision functionality specific to a particular technology/mechanism. As the rules to express, store and interpret policies can be different in different mechanisms, separate PDPs are required to determine policy decision specific to a particular mechanism. *e.g.* there can be

one PDP for access control lists, one for role based access control and another for SAML and XACML based access *etc*. As PDP is responsible for evaluating policy, it may require subject, resource, environment and other attributes for the evaluation of that policy. The attribute information is requested by PDP from authorization handler which in turn requests this information from PIP.

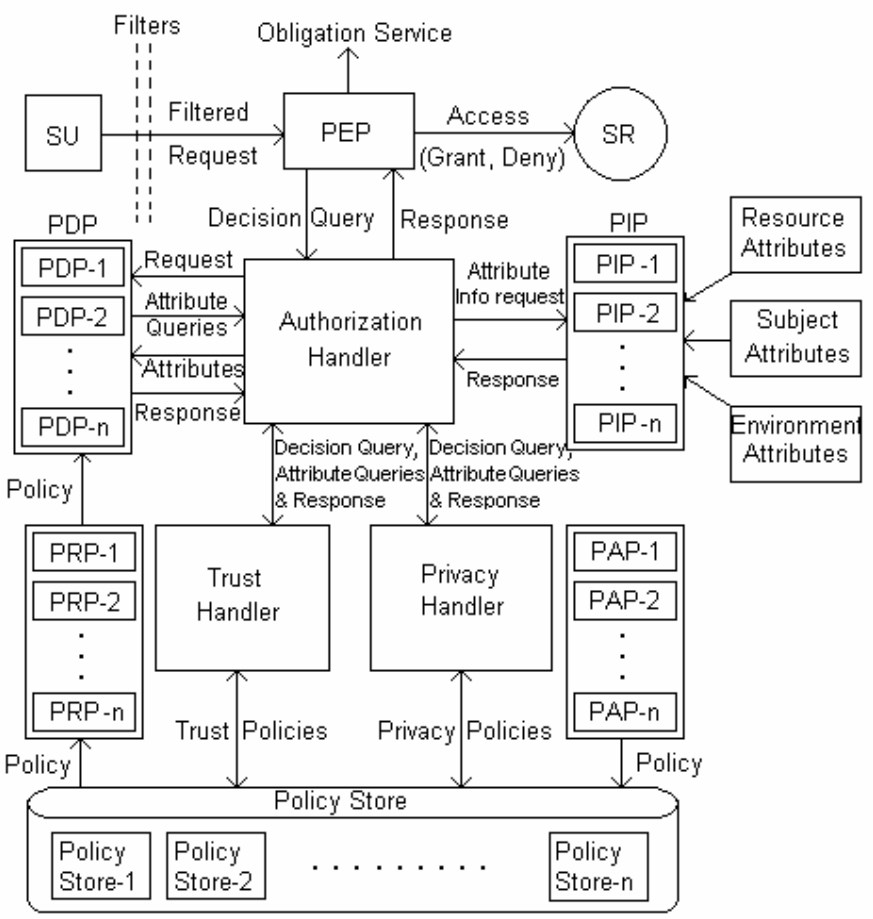

Figure 5.8: Integrated Policy Based Authorization Model

PIP (Policy Information Point) is used by authorization handler to retrieve subject, resource and environment related attributes. Like PDP, PIP is also proposed to be implemented as a combination of other PIPs (PIP-1, PIP-2, … PIP-n). As subject, resource, environment and other attributes might be stored in different formats at different places, separate mechanisms are required to fetch, understand and supply those attributes to authorization handler. PIP passes attribute information to authorization handler which in turn passes this information to PDP. Now PDP evaluate the policy and prepare the

result which is then passed to authorization handler. Trust handler and privacy handler provide trust and privacy based access information to authorization handler by evaluating trust and privacy relationships and policies. The authentication handler functionality has been implemented in PDP itself. After getting all the information from PDP, trust handler and privacy handler, authorization handler prepares a final result and passes it to PEP. Based on the result, PEP then grants/denies access to the requested service/resource. Obligation service, if any, is also executed by PEP.

5.4.3 High Level View of Implementation

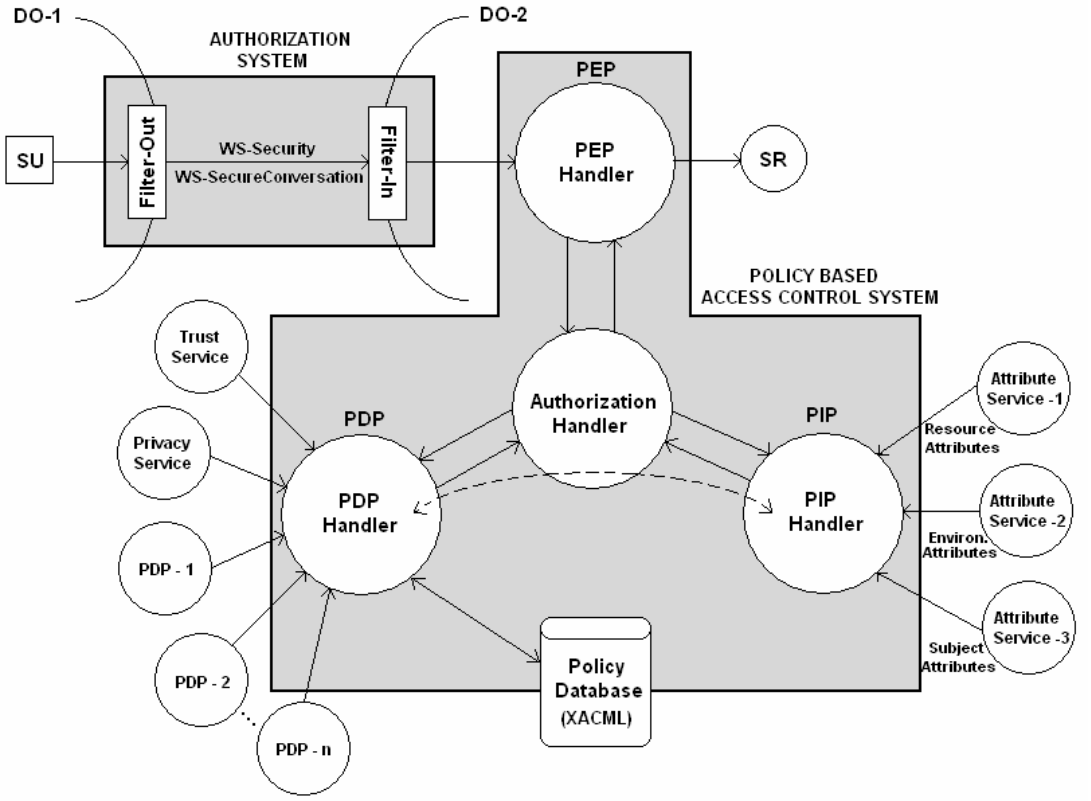

Figure 5.9: Schematic showing high level view of the implementation

Figure 5.9 shows a high level view of the implementation. As shown in Figure 5.9, subject's access request for Service SR first passes through Filter-Out component at the source domain and then through Filter-In component at the target domain. During this passage, subject's access rights are filtered for the target domain. At the target domain, PEP prepares authorization decision query and passes it to authorization handler. PEP,

PDP and PIP have been implemented as web services. Privacy and trust handlers have also been implemented as trust and privacy services. PDP handler makes use of trust and privacy services to get trust and privacy related access control information. It also gets information from other PDP implementations (PDP-1, PDP-2 … PDP-n). PDP makes use of policy database to fetch policies applicable to a particular access request. These policies are evaluated and evaluation result is prepared. PIP obtains attributes related to different aspects using attribute services and passes these to authorization handler. Authorization handler then passes these attributes to PDP for policy evaluation. Thus attributes required for evaluation come to PDP from PIP through authorization handler. This interaction is shown as dotted curved line between PDP handler and PIP handler in Figure 5.9. PDP prepares evaluation result and passes it to authorization handler. After getting all the information, authorization handler prepares final result which is then passed to PEP. Based on the response received from authorization handler, PEP either grants or denies access to requested service/resource. During all these steps the relevant information is stored in log tables also to address auditing and accounting requirements.

5.4.4 Services defined, implemented and exposed by Integrated Policy Based Authorization Model

Integrated Policy-based authorization model has been implemented as a collection of authorization and policy related security services exposed as web services. Some of the services defined, exposed and used in the implementation are:

Service Name	**Purpose**
getDomainPolicy( )	To get policy associated with a domain.
getServiceProviderPolicy ( )	To get policy associated with a service provider.
getServicePolicy( )	To get policy associated with a service.
authenticate( )	To authenticate a subject. This service is overloaded.
evaluateAuthenticationPolicy( )	To evaluate authentication policy associated with a service.
evaluatePrivacyPolicy( )	To evaluate privacy policy associated with a

	service.
evaluateTrustPolicy( )	To evaluate trust policy associated with a service.
evaluateAuthorizationPolicy( )	To evaluate authorization policy associated with a service.
evaluateSecurityPolicy( )	To evaluate security policy associated with a service.
getSourceDomain( )	To get source domain from request.
getSourceServiceProvider( )	To get source service provider from request.
getSourceSubject( )	To get source subject from request.
getAccessContext( )	To get context associated with the access request.
getSubjectAttributes( )	To get attributes associated with a subject.
getResourceAttributes( )	To get attributes associated with a service/resource.
getAccessibleDomains( )	To get list of accessible domains.
getBlackListedDomains( )	To get list of black listed domains.
getXACMLPolicy( )	To retrieve “policy” element from XACML policy.
getXACMLRule( )	To retrieve “rule” element from XACML policy.
getXACMLTarget( )	To retrieve “target” element from XACML policy.
setServicePolicy( )	To set policy associated with a service.
setServiceProviderPolicy ( )	To set policy associated with a service provider.
setDomainPolicy( )	To set policy associated with a domain.

Table 5.12: Services defined, implemented and exposed by integrated policy based authorization model

Most of the services listed above are overloaded and represent a subset of the total services exposed by integrated policy based authorization model.

5.5 Distinguished Features of Models

This section lists important features of the implemented models that make them different from other efforts in the relevant areas as described in the Chapter 2. Following paragraphs describe the important characteristics of each model.

The authentication model provides support for single sign-on and delegation features using proxy certificates and a credential management service to store, retrieve and update multiple user credentials. Following are the important characteristics of the authentication model.

- It provides credential repository and a service to manage multiple user credentials.
- It is based on Web Services Security specifications like WS-Security, WS-SecureConversation *etc.*
- It supports the storage of multiple types of tokens/credentials. (e.g. X.509, Kerberos, Custom Security Token *etc.)*
- It can be used for grid as well as web services.
- CMS is distributed over different domains, therefore removes the drawbacks of central storage.
- It provides support for single sign-on and delegation through proxy certificates.
- It provides a mechanism to express, evaluate and enforce authentication policies.

The privacy model provides support for anonymous access, hidden service access and access to private information based on conformance to privacy policies. Following are the characteristics of privacy model.

- It supports purpose based access to private information/resources.
- It supports hidden service access through custom security tokens.
- It supports anonymous access through the concept of anonymous security tokens.
- It provides mechanisms to handle privacy requirements of both, the service requesters as well as service providers.
- It provides mechanisms to express, evaluate and enforce privacy policies between service providers and requesters.
- It describes the integration of privacy based access with authorization framework.
- It uses XACML to express access control privacy policies which is OASIS standard.

The trust model provides support for calculating direct as well as recommended trust to determine trustworthiness of target service for enabling trust based access to grid services. Following are the important characteristics of the trust model.

- It can deal with identity as well as behavior trust.
- It provides support to calculate direct as well as recommended trust.
- It provides mechanisms to express, evaluate, and enforce trust policies.
- It describes the integration of trust based access with authorization framework.
- It also supports updation and management of trust relationships.

Integrated policy based authorization model provides policy based access to grid services and integrates with privacy and trust models by implementing trust and privacy based access. It provides support to express, evaluate and enforce different types of security policies (authentication, privacy, trust and authorization policies). It enables us to treat and specify the policy involving combination of authentication, privacy, trust and authorization related aspects as a single unit. The framework is flexible, scalable, supports fine grained access to services/resources and is able to express and enforce VO wide and other access control security policies. Following are the important characteristics of the integrated policy based authorization model.

- It supports fine grained and context based access to grid services/resources.
- Policy expression is platform independent.
- It is able to express and enforce VO wide and service wide access control policies.
- It introduces Filter components which make it flexible and scalable.
- It extends basic authorization mechanism to include trust and privacy based access to grid services.
- It supports multiple security policies and provides facilities to integrate different authorization mechanisms.

All these features make the implemented framework distinguished from other efforts. Next Chapter describes the implementation results and performance analysis of authentication, privacy, trust and authorization policies. The framework has been evaluated by implementing various security related scenarios and through different implementations that involve enforcement of different types of access control policies. The implementation has been done in .NET environment with the support of WSE 3.0 toolkit.

Chapter 6

Implementation Results and Performance Analysis

This chapter discusses the important security related scenarios that have been implemented through the security policy framework and also presents performance analysis of the individual authentication, privacy, trust and authorization policies. The implementation has been done in .NET environment with the support of WSE 3.0 toolkit. The implemented security services have been exposed as web services. Today the world is witnessing the convergence of grid and web services. The two (grid and web) started far apart in specifications, technology and applications but now they are converging into a common set of standards and specifications. So security services have been implemented as web services. WSE 3.0 implements many web services security specifications like WS-Security, WS-SecureConversation, WS-Trust *etc.* We have made use of these specifications to implement the framework. For other specifications like XACML and SAML, we are not using any software instead the necessary functionality has been developed and used wherever required. All security information is exchanged as SOAP messages. SOAP messages are constructed and WS-Security information is embedded using WSE 3.0 toolkit. The other web services security specifications like WS-Trust and WS-SecureConversation have been used for security token exchange and to establish secure communication contexts. Using web services security specifications, we are also guarantying the confidentiality and integrity of the messages exchanged as WSE 3.0 provides features for encrypting selected parts of SOAP messages as well as digital signatures. All types of security policies have been expressed in simple XML or XACML and are stored in the policy database. The XML database has been developed in MS-SQL Server. To implement specific aspects of the security policy framework, we are making extended use of XACML and other specifications.

For implementation purpose, the .NET based approach has been chosen to increase the participation of .NET programmers and communities in the development and use of grid applications. Though most of the grid applications and infrastructures in use today are based on Unix/Linux platform but initiatives in the direction of .NET based approaches to grid application development are emerging. *e.g.* Alchemi [93] is a desktop grid framework based on .NET. The caBIG (cancer Biomedical Informatics Grid) [157] deals with creating a design to support caBIG and caGrid [158] (underlying platform that provides the basis for connectivity of caBIG tools) using .NET. NGrid [159] is an open source grid computing framework written in c#. GridFTP [160] is a data transfer protocol for accessing distributed data on the Grid using .NET. WSRF.NET [161] is an implementation of full set of specifications for WSRF and WS-Notification on the .NET framework. OGSI.NET [162] is the implementation of Open Grid Services Infrastructure (OGSI) on .NET framework.

The .NET based implementation of grid security can complement the .NET approaches and projects described above. .NET based approach also supports rapid application development, involves less installation, configuration and development effort, is good GUI based and provide easy error handling and debugging facilities. The proposed models are not platform/hardware dependent. The proposed models can be implemented on UNIX based systems as well. Following sections detail about scenarios implementation and performance analysis.

6.1 Scenarios Implementation

This section presents the important security related scenarios that have been implemented through the security policy framework. The support for these scenarios is generally required in most of the grid and web based systems. Microsoft and IBM in their joint paper "Security in Web Services World: A Proposed Architecture and Roadmap" [33] have described many important security related scenarios that must be addressed by a security implementation. The security policy framework supports all these scenarios. We have divided the implemented scenarios into four categories namely authentication, privacy, trust and authorization related scenarios. Following paragraphs discuss each of these categories.

6.1.1 Scenarios related to Authentication

The security policy framework is capable of implementing single sign-on and delegation along with non-repudiation, confidentiality, integrity of exchanged information and secure conversation. Single sign-on and delegation features are supported through proxy certificates. The other security requirements like non-repudiation, confidentiality, integrity of exchanged information and secure conversations are supported through XML-Signature, XML-Encryption, WS-Security and WS-SecureConversation specifications. WSE 3.0 toolkit under .NET environment has been used to incorporate the functionality of WS-* specifications in implemented scenarios. Figure 6.1 and Figure 6.2 represent single sign-on and delegation scenarios.

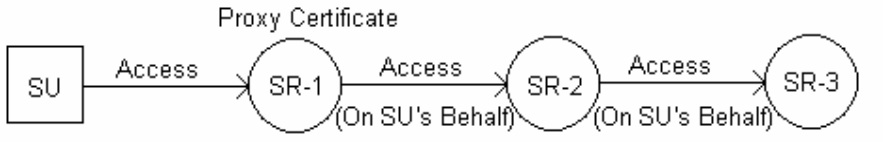

Figure 6.1: Schematic showing Single Sign-On

In Figure 6.1, Subject SU issues proxy certificate to Service SR-1 so that SR-1 can act on its behalf. SR-1 uses this proxy certificate to get authentication at SR-2 on SU's behalf. Similarly, using the same concept, Subject SU also gets authenticated at SR-3 by SR-2. To generate proxy certificates, the sequence of steps described in Table 5.1 have been used.

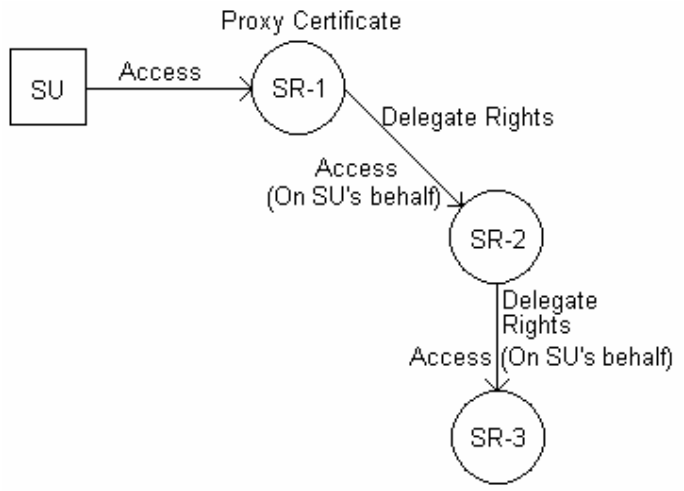

Figure 6.2: Schematic showing Delegation and Single Sign-On

Figure 6.2 represents delegation scenario. Here Subject SU passes some/all of his access rights to SR-1 using proxy certificates. The delegated rights are embedded in proxy certificates through PCI (Policy Certificate Information) field. In this case SR-1 can access SR-2 on SU's behalf but only the actions implied by delegated rights can be performed. The delegated rights can further be delegated if the original subject has passed the right of further delegation in the delegation list.

6.1.2 Scenarios related to Privacy

The security policy framework is capable of implementing anonymous access and privacy based access to services/resources. It also supports the access of hidden services through custom security credentials. Figure 6.3, Figure 6.4 and Figure 6.5 represent scenarios related to access of private information, anonymous access and hidden services access, which have been implemented through the framework.

In Figure 6.3, Subject SU sends his private information to Service SR-1 and both establish and agree on a privacy relationship. If SR-1 exposes SU's private information then access to this information is provided only if the requester (in this case SR-2) conforms to established privacy policies and relationships. To implement this scenario, the privacy based access mechanism as presented in Table 5.4 has been used.

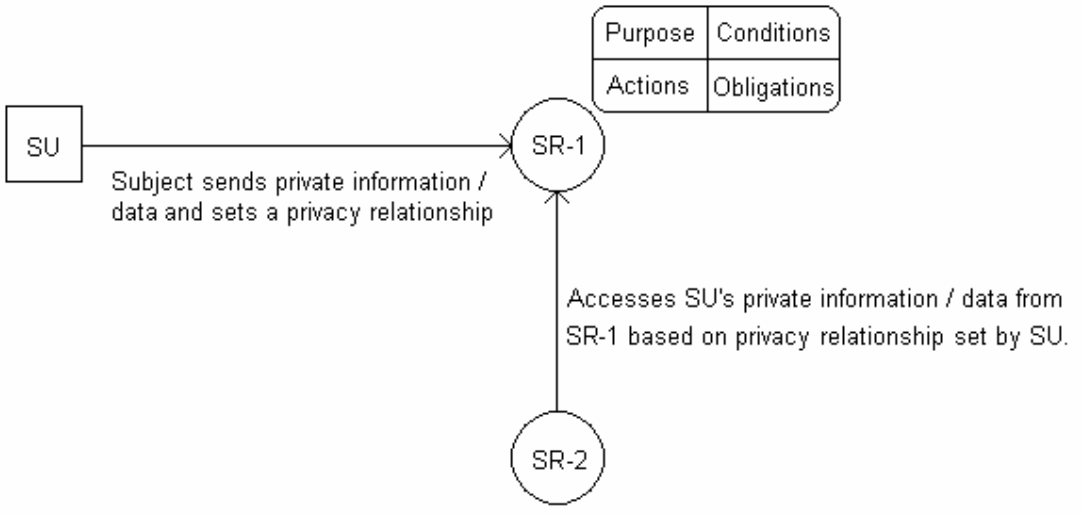

Figure 6.3: Schematic showing access of private information based on established privacy policies and relationships

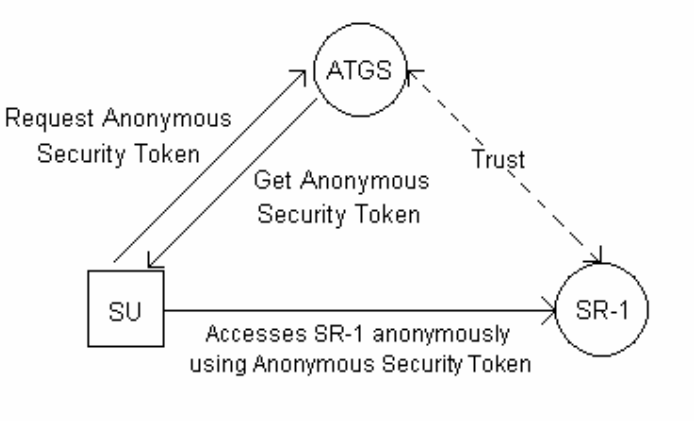

Figure 6.4: Schematic showing anonymous access of a service

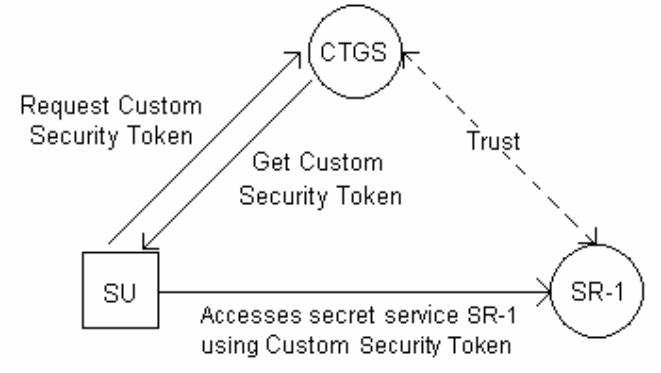

Figure 6.5: Schematic showing access of a hidden service

Figure 6.4 shows anonymous access of a service. In this case, Subject SU requests anonymous security token from an ATGS (Anonymous Security Token Granting Service) and presents it to service SR-1 in order to access it anonymously. Here a prior trust relationship exists between ATGS and SR-1. The anonymous security token asserts about authorization information of the subject but removes its identity. Thus subject can access the service anonymously.

Figure 6.5 shows how access to a secret service is provided. In this case SU requests custom security token from Custom Security Token Granting Service (CTGS) and presents it to SR-1 to access it. As custom security token is generated by CTGS, SU has no idea of the information present in custom security token. SU does not know which of his attributes / other information is embedded in this token. SU simply presents the custom security token to SR-1 and access the service but does not know exactly which of his attributes/credentials are causing access to this service. Thus SU can access the

service without knowing the exact security requirements of the service. The anonymous security tokens and custom security tokens have been generated through tools available in .NET environment.

6.1.3 Scenarios related to Trust

The security policy framework is capable of determining the trustworthiness of target service/resource through direct as well as recommended trust. Figure 6.6 and Figure 6.7 show scenarios related to direct and recommended trust which have been implemented.

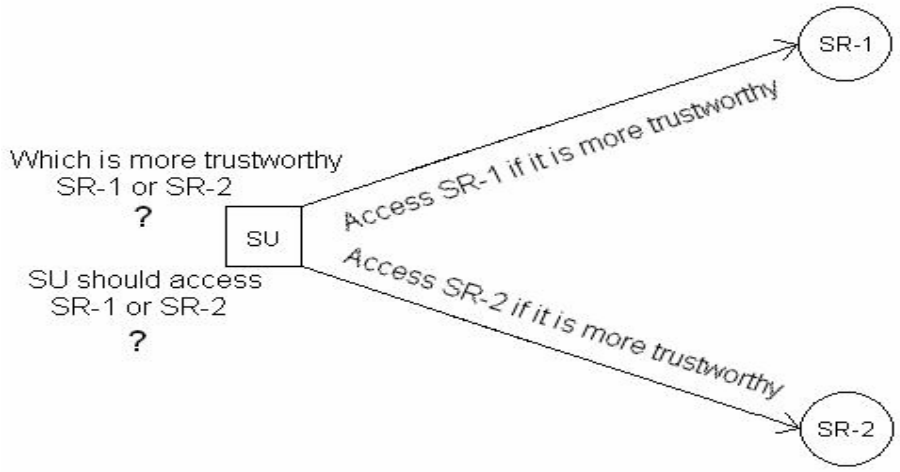

Figure 6.6: Schematic showing determining direct trust with a service

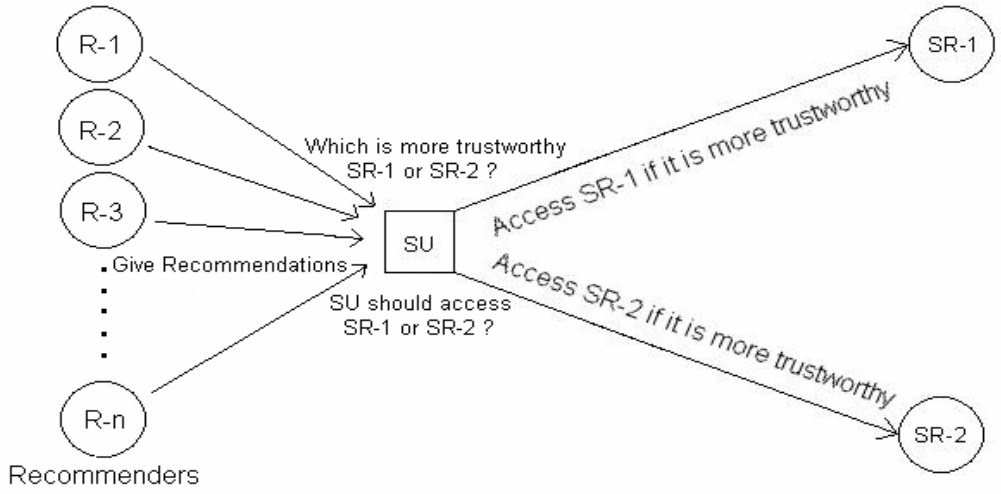

Figure 6.7: Schematic showing determining direct and recommended trust with a service

In Figure 6.6, Subject SU determines the trustworthiness of different services (in this case SR-1 and SR-2) exposed in the environment and based on the result accesses a particular service which comes out to be more trustworthy. In determining trust on a service, subject can also take help from his recommenders. This is shown in Figure 6.7. After determining the direct and recommended trust, the subject makes a final decision regarding which service to access. To calculate direct and recommended trust, the pseudo codes presented in Table 5.8 and Table 5.9 have been used. The framework implements these trust related scenarios. Different variations of these scenarios can also be implemented through the framework using other services (listed in Table 5.10), exposed by the trust model.

6.1.4 Scenarios related to Authorization

The security policy framework is capable of implementing distributed access and policy based access. Figure 6.8, Figure 6.9 and Figure 6.10 represent scenarios related to policy based access, distributed authorization and federation of security services that have been implemented through the framework.

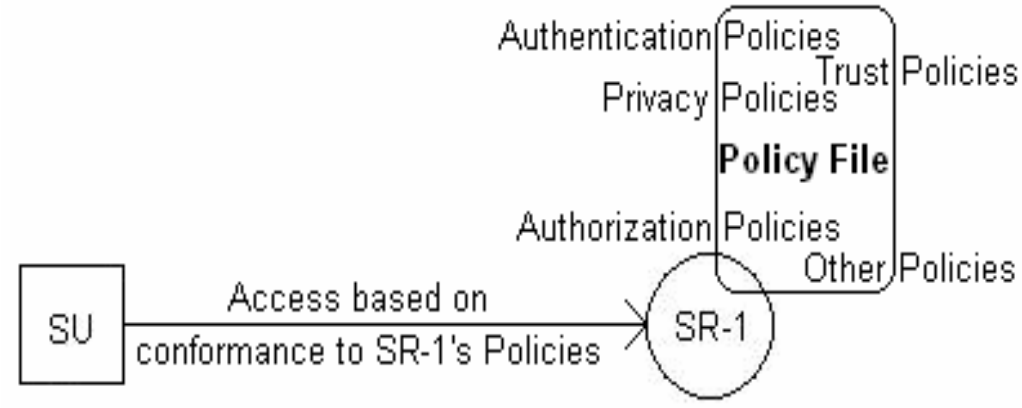

Figure 6.8: Schematic showing access based on conformance to service's policies

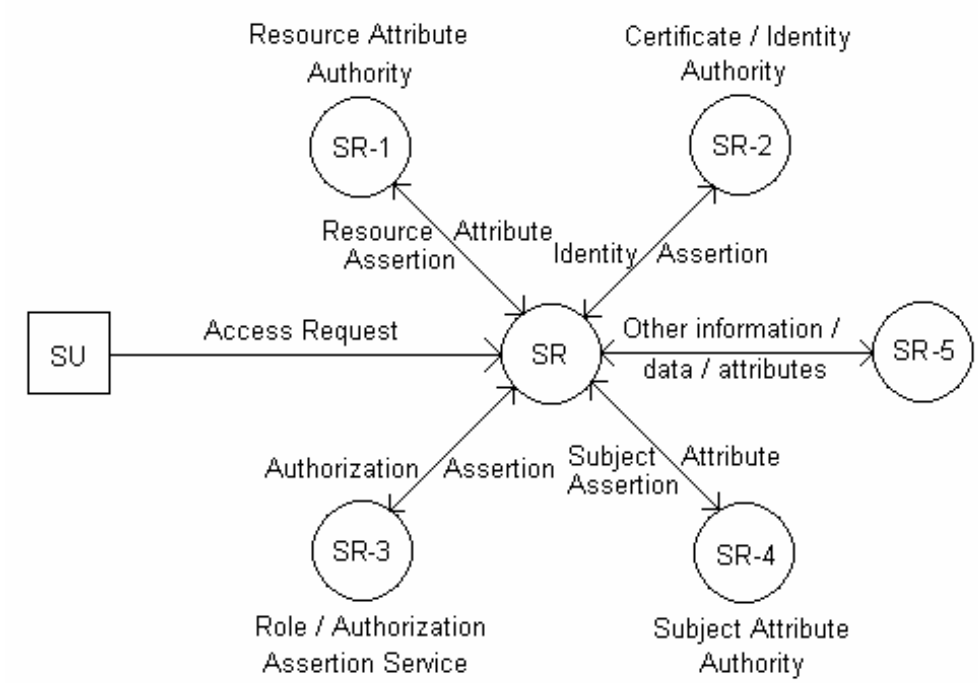

Figure 6.9: Schematic showing scenario related to distributed authorization

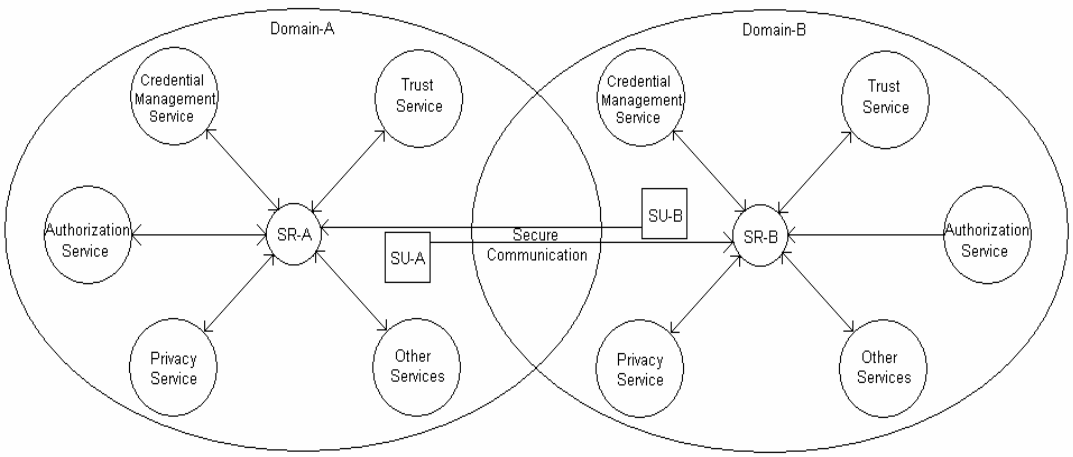

Figure 6.10: Schematic showing scenario related to federation of security services

In Figure 6.8, Subject SU generates access request to access service SR-1. Access to this service is provided based on conformance to its policies. To implement this scenario, the sequence of steps presented in Table 5.11 have been used. Figure 6.9 represents distributed authorization. To determine authorization information and attributes, a service

takes the help of other services. The service receives identity assertions, subject and resource attribute assertions and other information/data from different authorities distributed in the environment. These assertions are used to prepare final authorization result. Figure 6.10 shows the federation of security services. Subject of one domain access the services of other domains which are protected by security services of their own domain. All these scenarios have been implemented successfully. The security policy framework is also capable of implementing variations of these scenarios.

6.2 Performance Analysis

This section presents the performance analysis of individual authentication, privacy, trust and authorization policies. The implementation environment consists of 50 domains with users ranging from 10 to 50 in each domain. All the domains have more than 10 service providers that provide different services/resources to other domains. Resources have been exposed as services. Subjects have been given different security credentials (X.509 certificate, username: password, Kerberos *etc.*). These credentials have been generated through .NET tools. Database contains authentication, privacy, trust and other security policies established among subjects and services of different domains. Policy database is maintained by every domain for its subjects. Each service is associated with a policy file that describes its service policy. The security policies are exposed by services in the environment. Access to services/resources is provided after conformance to these policies. For the performance analysis of different policies, we have created a sample service and attached to it the policies related to the type (authentication, privacy, trust and authorization) which is being analyzed. For each different type, we have noted the time taken by the framework in evaluating individual policy of that type and then the set of policies of that type by varying the number of policies in the policy file.

Time taken by different components *i.e.* PIP, PDP and PEP in evaluating each type of policy has been noted separately. Time taken by PIP component represent the time it takes to fetch, interpret and supply subject, resource, environment and other attributes to PDP for the evaluation of a policy. Time taken by PDP component represent the time taken to fetch, interpret and evaluate applicable policy. Note that PDP time includes PIP time also. Time taken by PEP component represent the time to prepare authorization decision query, passing it to PDP and receiving the response. Thus it includes PDP time

also and represents the overall time taken to enforce a policy. As PEP's task is to enforce policy according to response received from PDP and PDP being the major component for PEP in evaluating policy, there is not much difference in PDP time and PEP time. Another thing to note is that the function of privacy handler and trust handler is also like PDP component as in the framework, they are being used to evaluate privacy and trust policies specifically. This is clear from Figure 5.9 also where privacy and trust handler components have been shown as privacy and trust services parallel to different PDP services. In the following paragraphs, for all types of policies, we have used the term PDP to represent the component responsible for evaluating policies.

Different implementations have been tested on systems of different configurations, with memory ranging from 256MB to 1GB and processor speed ranging from 900MHz to 2GHz, and same result patterns have been observed. Moreover the execution has also been performed by varying the number of entities (domains, service providers, services, subjects *etc.*). In this case also the same result patterns have been observed. Graphs of following sections shows the result of implementation on most constrained system *i.e.* system having 256MB of RAM and 900MHz of CPU. The number of entities (domains, service providers, services, subjects *etc.*) used in the implementation are same as mentioned in the beginning of this section. The comparison of different types of policies with each other has also been done with respect to time taken by them for their enforcement. Following sections present these details.

6.2.1 Performance analysis of authentication policies

For the performance analysis of authentication policies, 50 pure authentication related policies have been created and attached to the service. These policies cover different aspects of authentication like the security credentials required by the service, encryption and signature requirements, single sign-on and delegation requirements of the service *etc.* Then the time taken by each of these policies to get evaluated individually is noted. The time taken by PIP, PDP and PEP components is also noted separately. The graphs of Figure 6.11, Figure 6.12 and Figure 6.13 show these details. For authentication policies, evaluation functionality is provided by the authentication handler which has been implemented in PDP itself. The average time taken by PEP component for authentication policies comes out to be 53.38 ms. There is not much difference in time taken by PEP

component to evaluate different authentication policies. This is because different types of authentication policies involve similar type of processing and almost all the information required to evaluate authentication policy is available from the access request itself. The resource, environment and other attribute access requirements in evaluating authentication policies are minimal. This is clear from the PIP time also as shown in Figure 6.11, which comes out to be 0.32 ms.

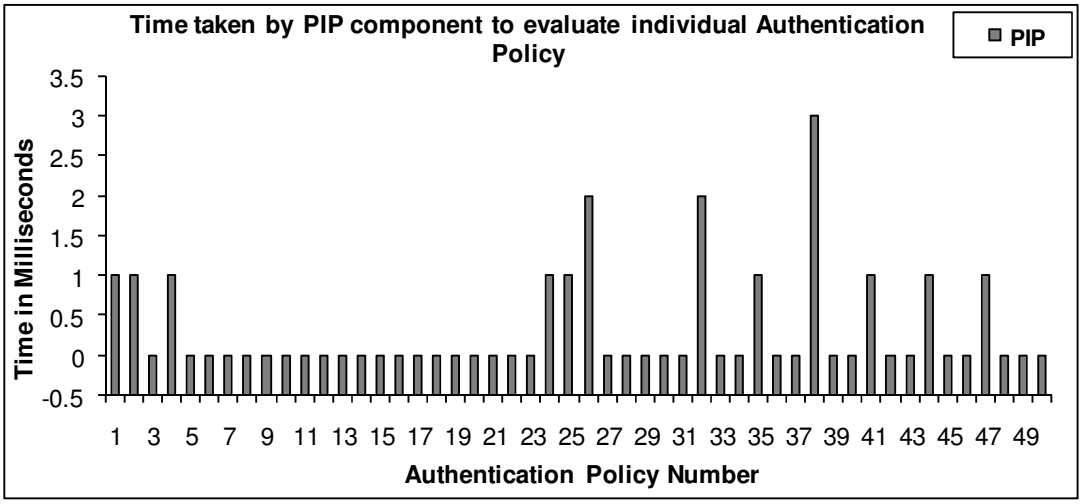

Figure 6.11: PIP time to evaluate individual authentication policy

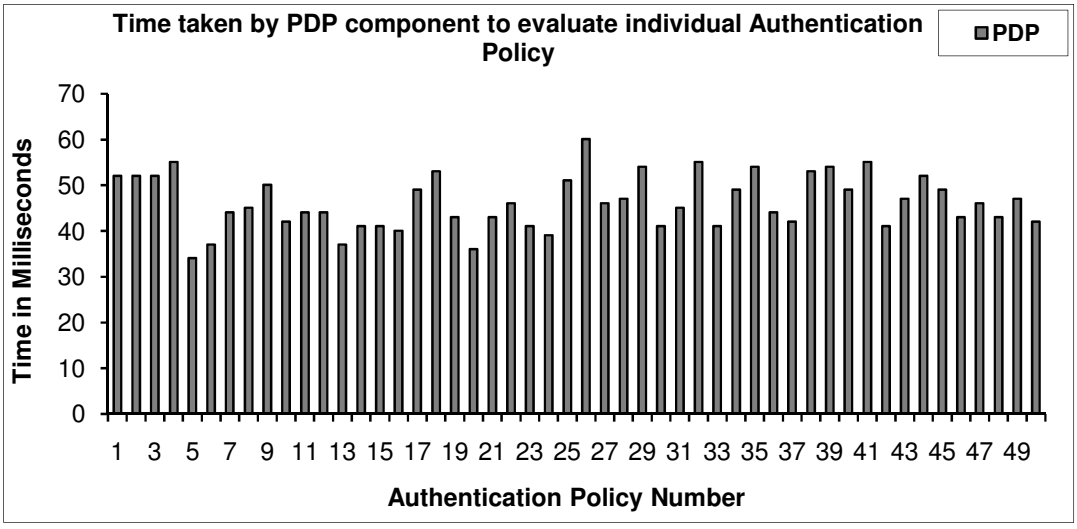

Figure 6.12: PDP time to evaluate individual authentication policy

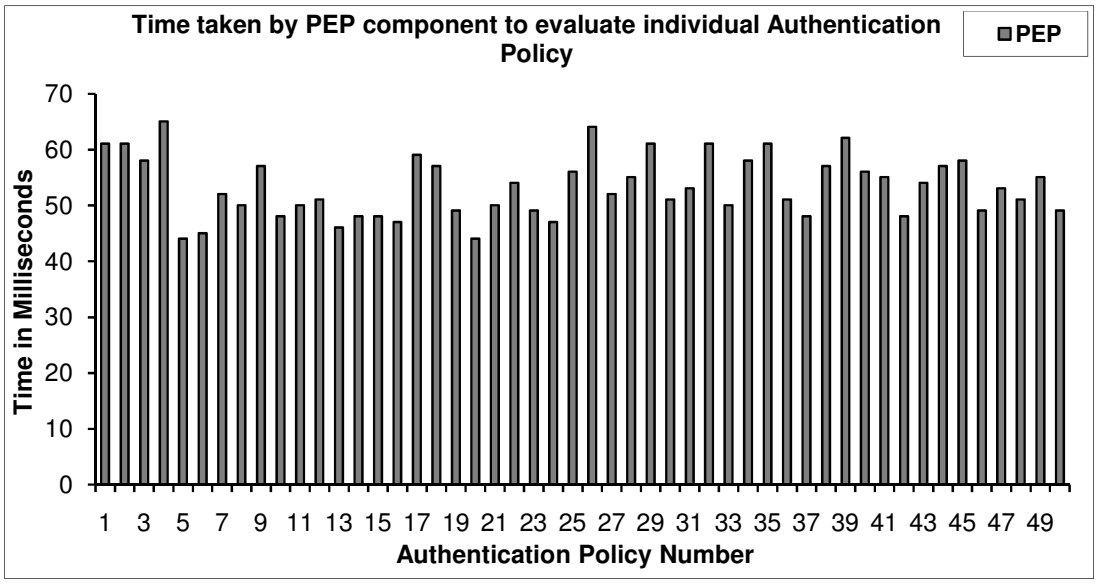

Figure 6.13: PEP time to evaluate individual authentication policy

In the second phase, the number of authentication policies attached with the service have gradually been increased and the time taken by PIP, PDP and PEP components is noted. Graphs of Figure 6.14, Figure 6.15 and Figure 6.16 show these details.

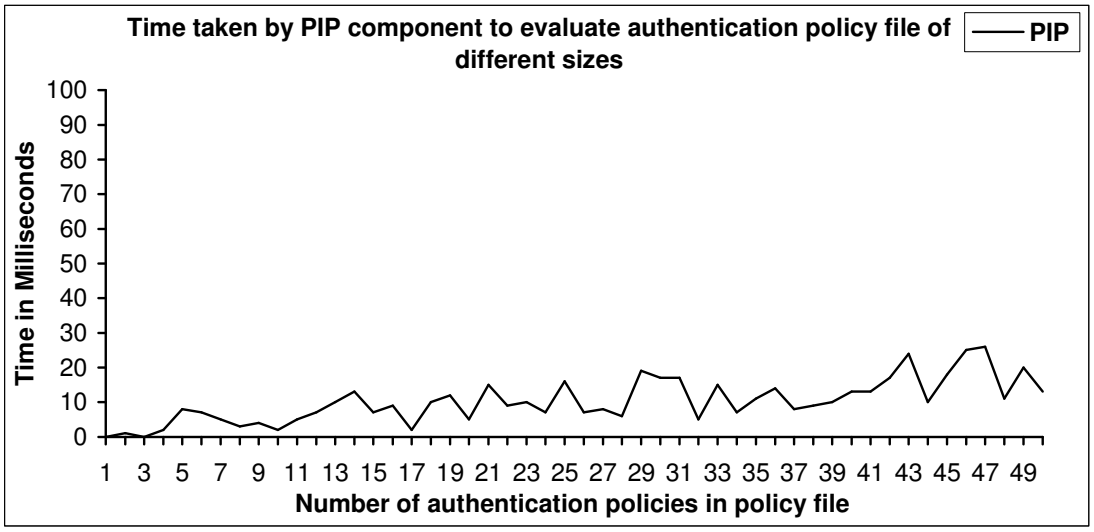

Figure 6.14: PIP time to evaluate authentication policy file of different sizes

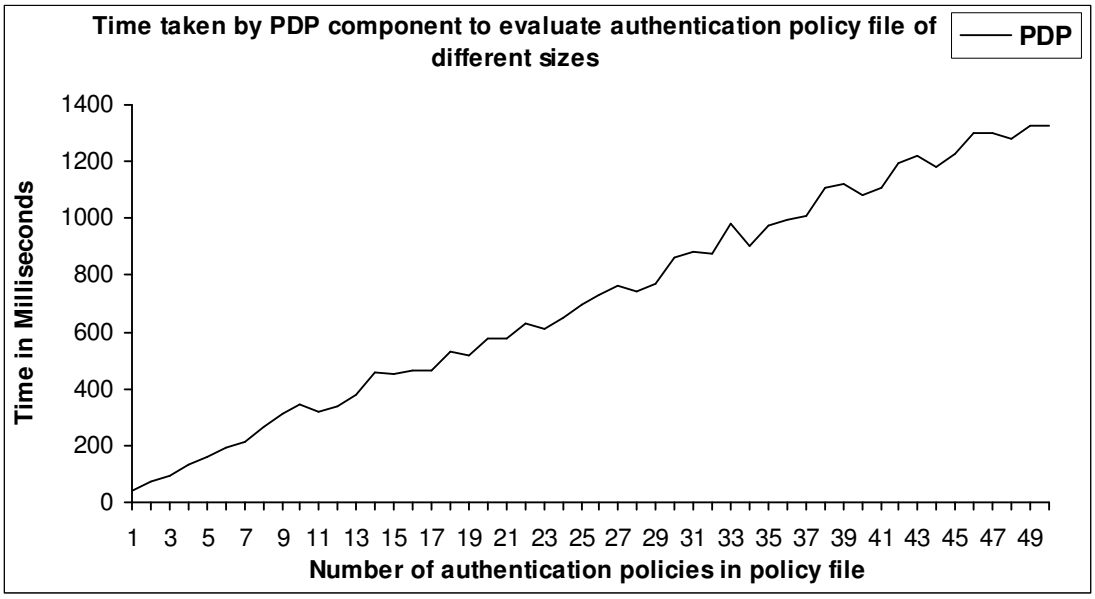

Figure 6.15: PDP time to evaluate authentication policy file of different sizes

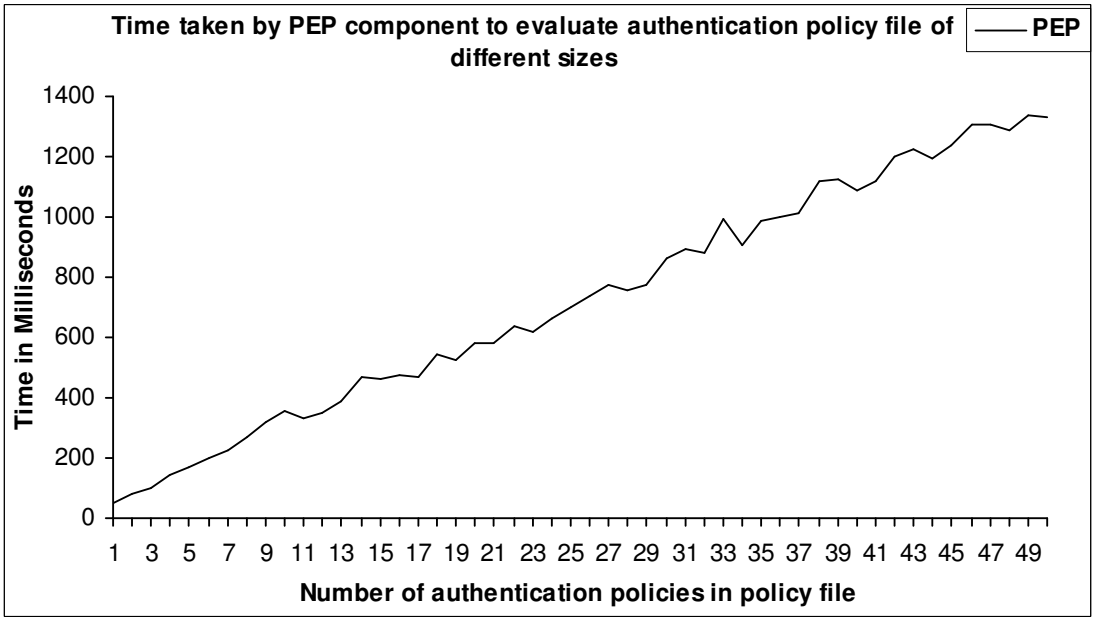

Figure 6.16: PEP time to evaluate authentication policy file of different sizes

The attribute access requirements in this case also remained minimal and did not increase significantly by increasing the number of authentication policies. On the other hand, the time taken by PEP and PDP components increased linearly with the increase in number of authentication policies. This is what we were also expecting. It is clear from

the graphs of Figure 6.14, Figure 6.15 and Figure 6.16 that increasing the number of authentication policies attached with a service does not adversely affect the performance of the authentication framework.

6.2.2 Performance analysis of privacy policies

For the performance analysis of privacy policies, 50 privacy related policies have been identified that cover different aspects related to privacy like purpose based access, hidden service access, anonymous service access, access of private information *etc.* Then the time taken by PIP, PDP and PEP components to evaluate individual privacy policy is noted. For privacy policies, PDP functionality is provided by privacy handler. The average time taken by PIP, PDP and PEP components for privacy policies comes out to be 24.65 ms, 73.58 ms and 81.34 ms respectively. Compared to authentication policies, more access to subject, resource and environment attributes is noted. This is because privacy based access involves processing of private information and policies that make use of these attributes. The details of time taken by different components are shown in the graphs of Figure 6.17, Figure 6.18 and Figure 6.19.

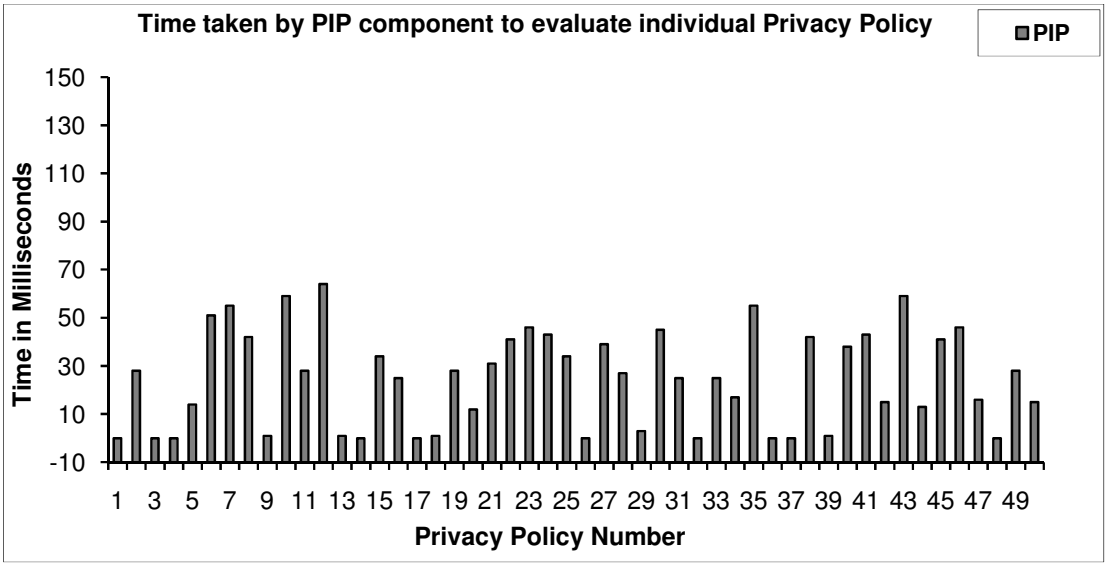

Figure 6.17: PIP time to evaluate individual privacy policy

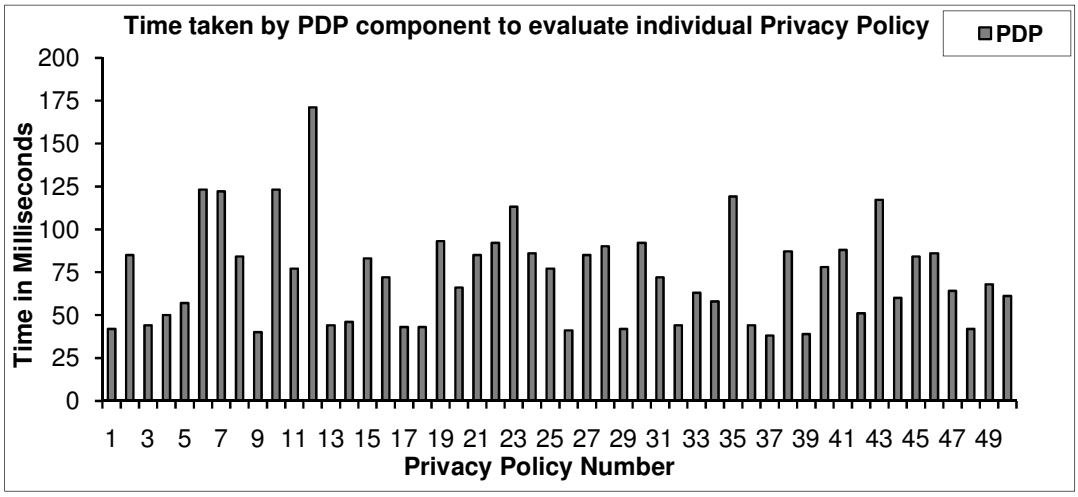

Figure 6.18: PDP time to evaluate individual privacy policy

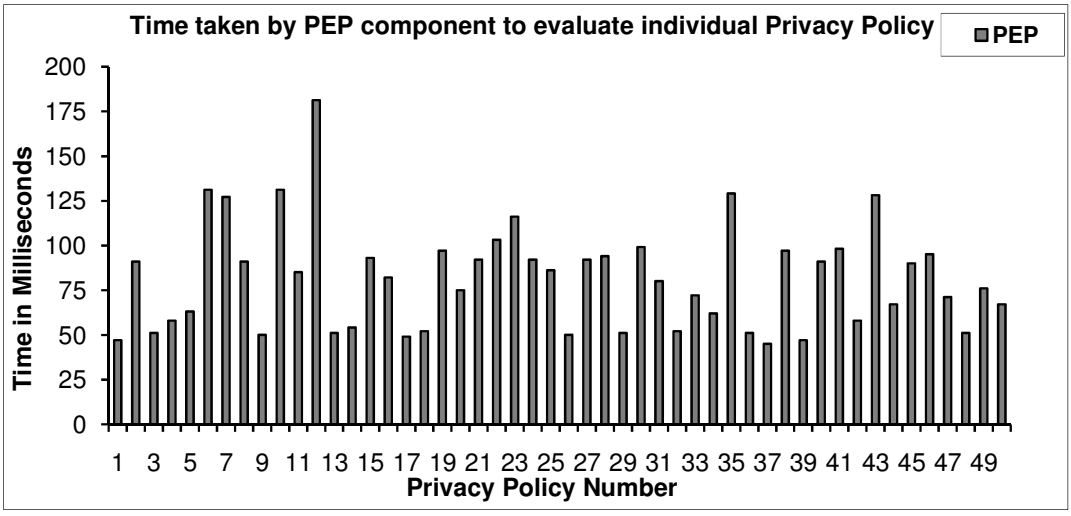

Figure 6.19: PEP time to evaluate individual privacy policy

Next the time taken by PIP, PDP and PEP components to evaluate different number of privacy policies is noted. The time taken by all the components increases linearly and not exponentially with the increase in number of privacy policies attached with the service/resource. This shows that privacy model also does not adversely affect the performance of authorization framework with the increase in number of privacy policies. The details are shown in the graphs of Figure 6.20, Figure 6.21 and Figure 6.22.

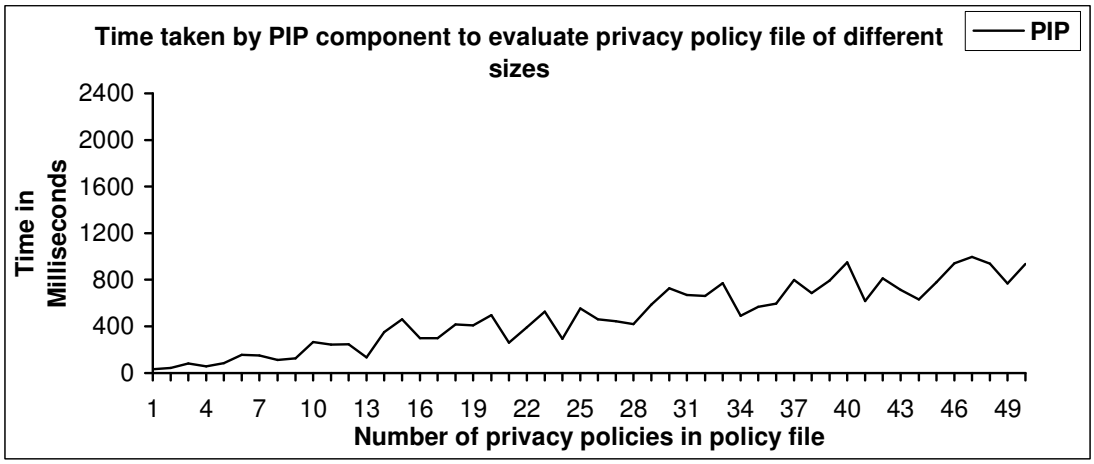

Figure 6.20: PIP time to evaluate privacy policy file of different sizes

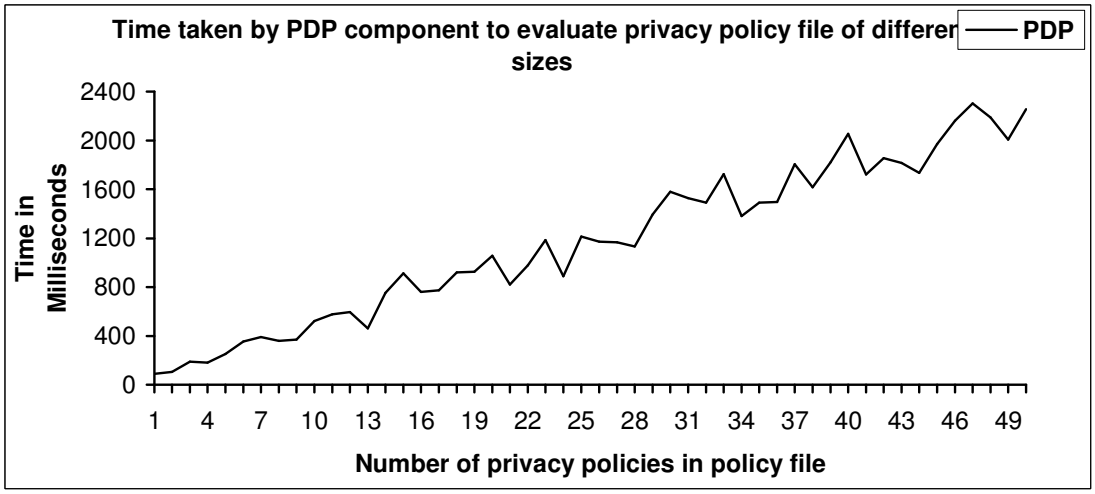

Figure 6.21: PDP time to evaluate privacy policy file of different sizes

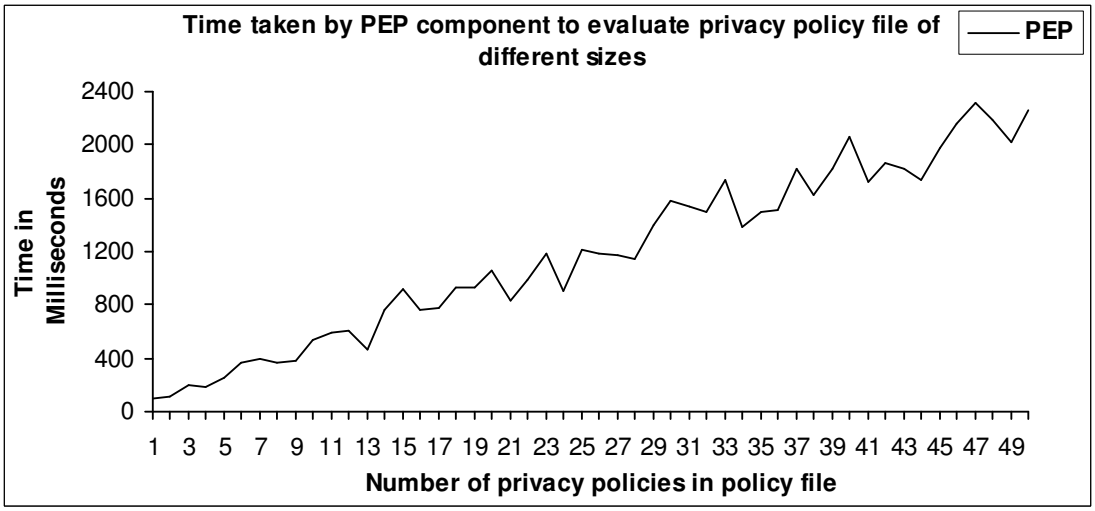

Figure 6.22: PEP time to evaluate privacy policy file of different sizes

6.2.3 Performance analysis of trust policies

Like authentication and privacy policies, for the performance analysis of trust policies, 50 trust related policies have been created. These policies include different aspects related to trust and involve determining trustworthiness of domain, service provider or service. Then these policies have been attached to the service and the time taken by PIP, PDP and PEP components for each trust policy is noted. The average time taken by PEP component in this case comes out to be 178.16 ms which is higher than the average time taken by PEP component for individual authentication and privacy policy. This is because the determination of trust value of target (service, service provider or domain) involves calculations and access to history of past interactions taken place between source and target. The average time taken by PIP component is also higher compared to time taken by PIP component for authentication and privacy policies as trust policies make more use of subject, resource, environment and other attributes. Graphs of Figure 6.23, Figure 6.24 and Figure 6.25 show these details.

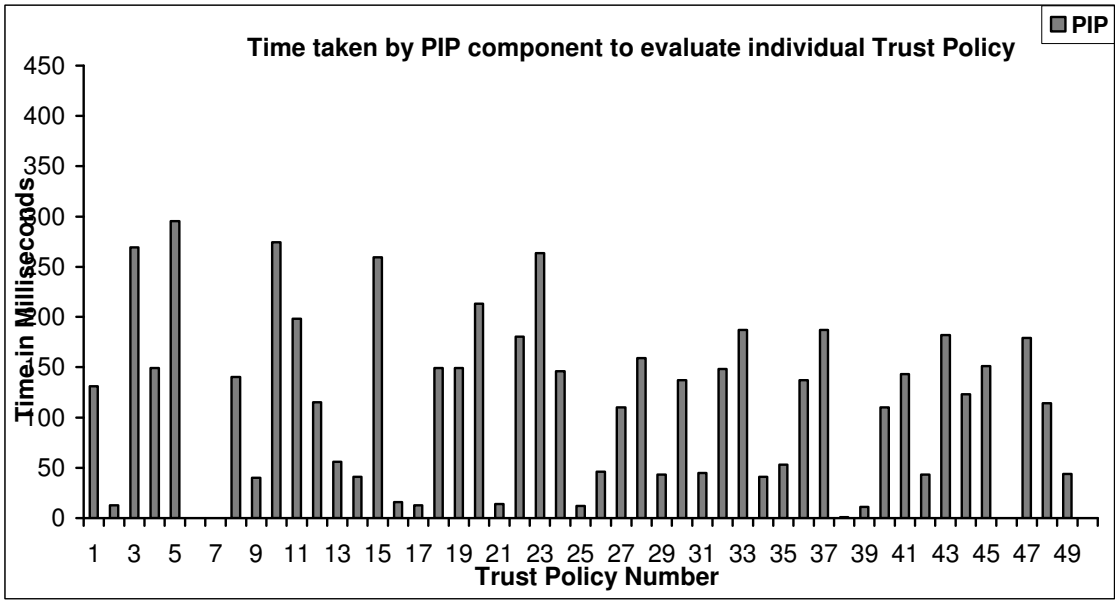

Figure 6.23: PIP time to evaluate individual trust policy

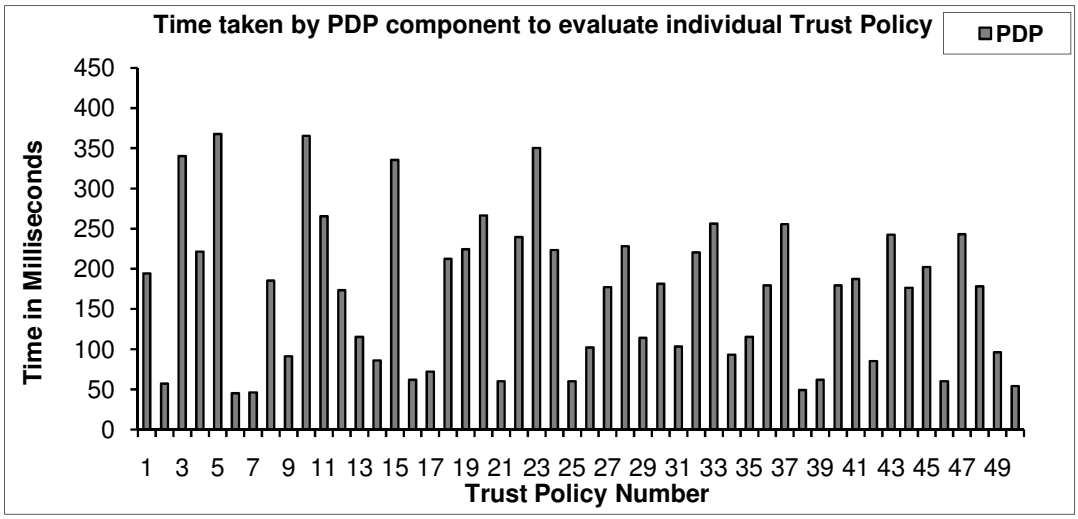

Figure 6.24: PDP time to evaluate individual trust policy

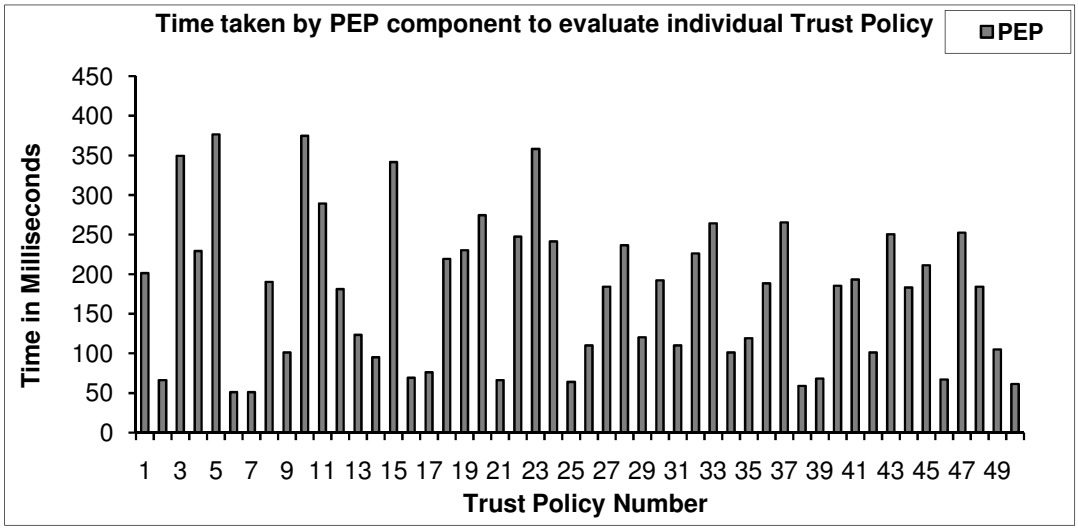

Figure 6.25: PEP time to evaluate individual trust policy

Next the time taken to evaluate different number of trust policies is noted. For this the number of trust policies attached with the service have been varied form 1 to 50 and the time taken by PIP, PDP and PEP components have been noted. The same behavior pattern as appeared with authentication and privacy policies is observed but in this case the PIP, PDP and PEP time increases more assertively with the increase in number of trust policies. This is again due to the fact that determination of trust value of a target involves calculations and access to history of past interactions taken place between source and

target, which is time consuming. Therefore, performance can be a concern if the number of trust policies attached with a service/resource are more. Graphs of Figure 6.26, Figure 6.27and Figure 6.28 show these details.

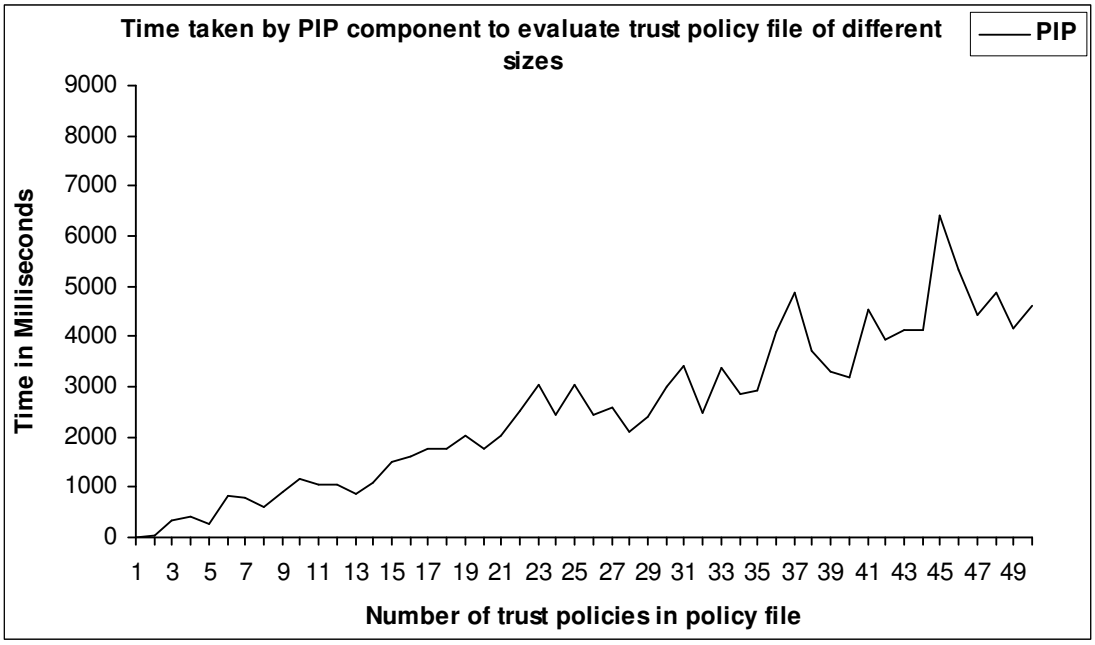

Figure 6.26: PIP time to evaluate trust policy file of different sizes

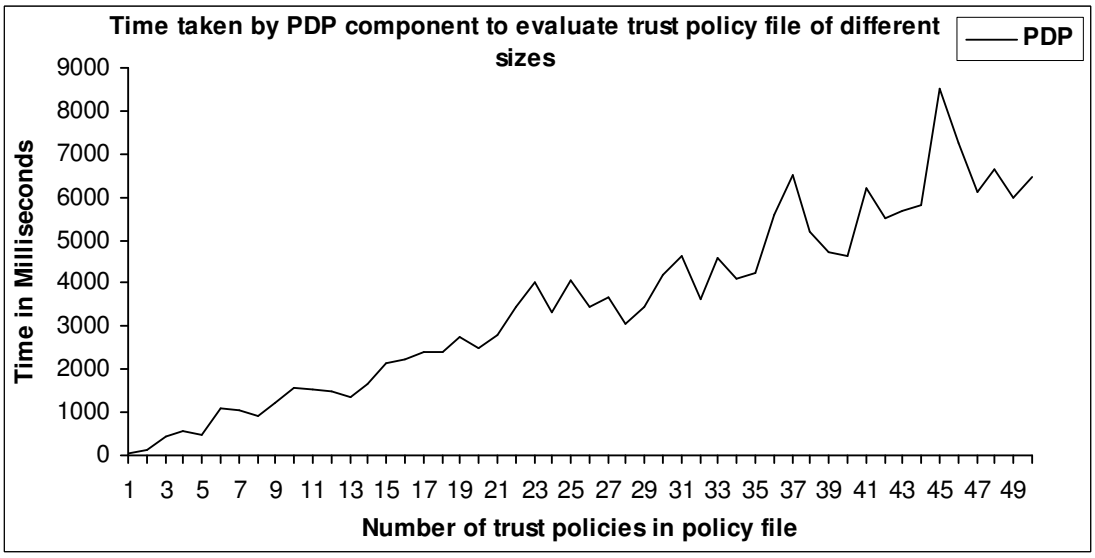

Figure 6.27: PDP time to evaluate trust policy file of different sizes

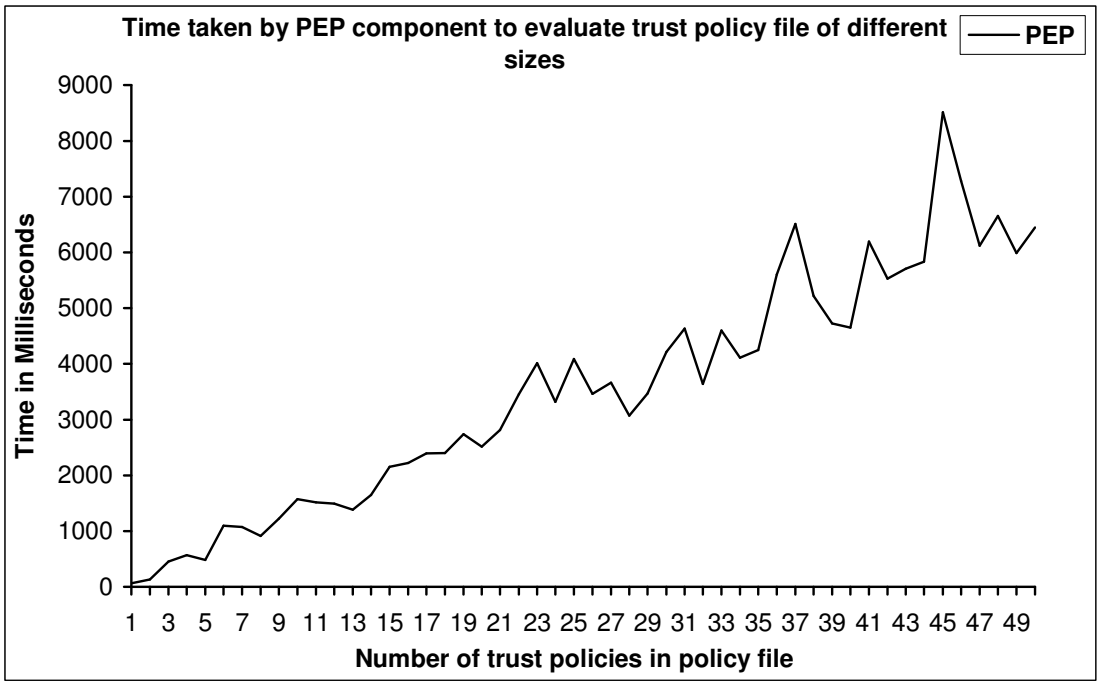

Figure 6.28: PEP time to evaluate trust policy file of different sizes

6.2.4 Performance analysis of authorization policies

The performance analysis of authorization policies involve attaching 50 pure authorization related policies with a service and noting the time taken by PIP, PDP and PEP components for their evaluation individually and accumulatively. These policies are chosen such that they do not involve any authentication, privacy and trust related aspects. The average time taken by PEP component for authorization policies comes out to be 83.58 ms which is significantly less than the average time taken by PEP component for trust policies, and more than that of authentication policies. This value is close to average time taken by PEP component for privacy policies. Also the average time taken by PIP component is 22.55 ms compared to 0.32 ms in case of authentication policies, 24.65 ms in case of privacy policies and 110.74 ms in case of trust policies. The graphs of Figure 6.29, Figure 6.30 and Figure 6.31 show these details.

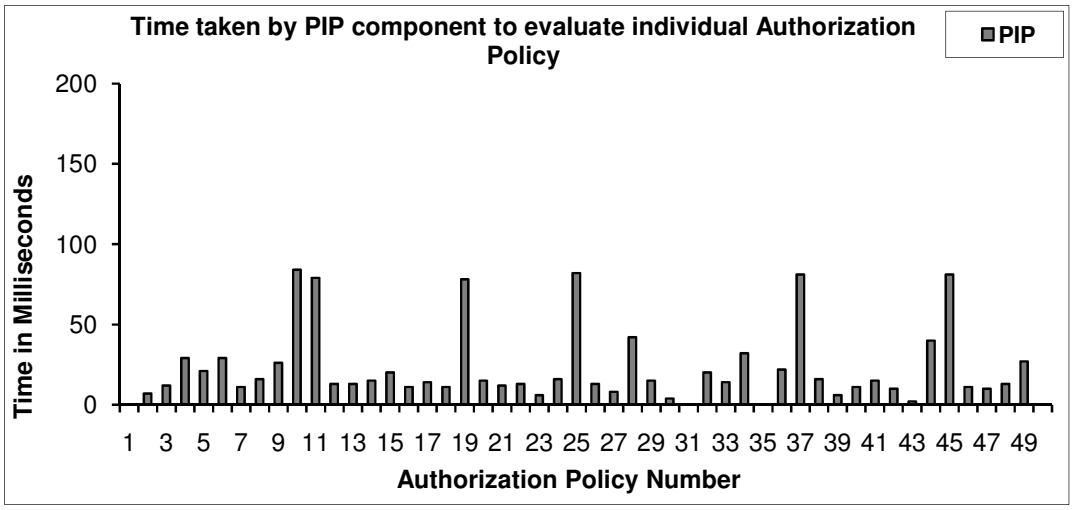

Figure 6.29: PIP time to evaluate individual authorization policy

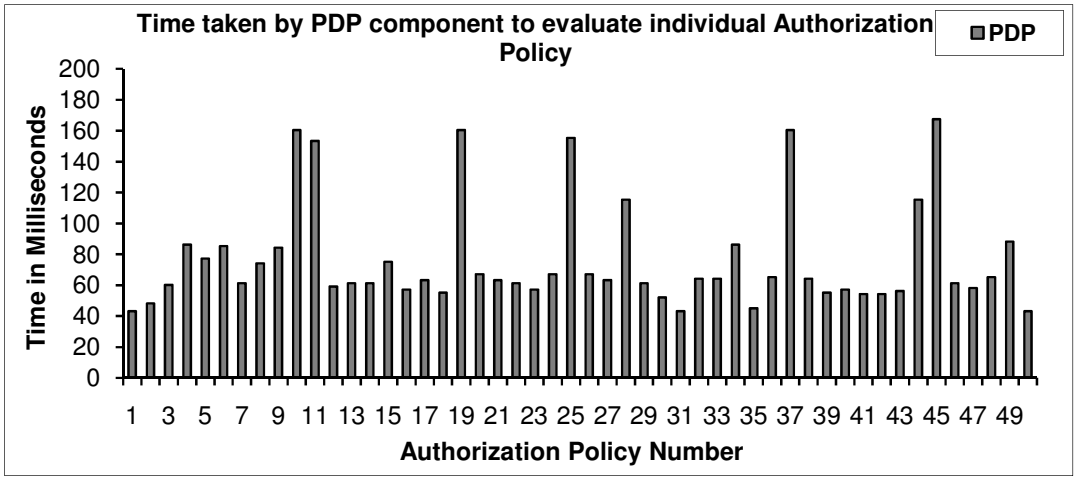

Figure 6.30: PDP time to evaluate individual authorization policy

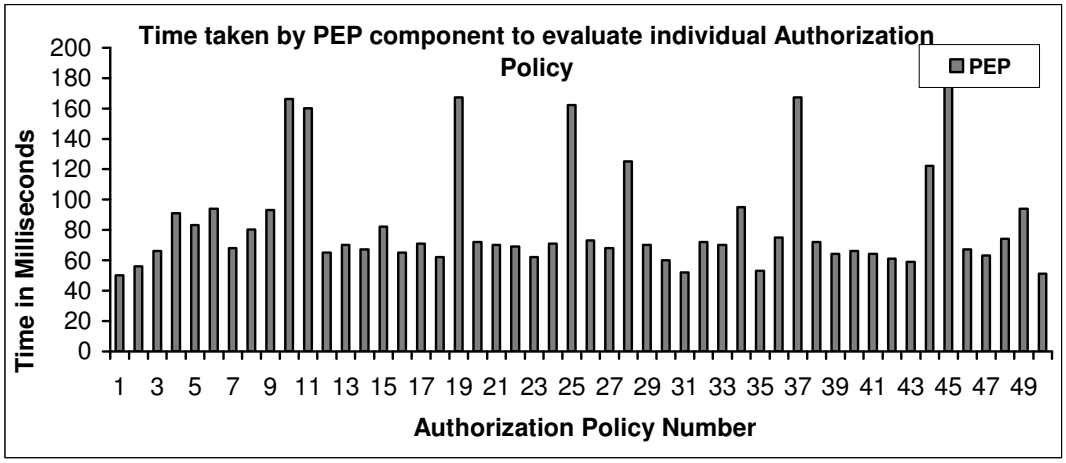

Figure 6.31: PEP time to evaluate individual authorization policy

Figure 6.33 and Figure 6.34 show how the time taken by PIP, PDP and PEP components increases with the increase in number of authorization policies. From these graphs, it is clear that the time taken by all the components increases linearly and not exponentially with the increase in number of authorization policies. Thus increasing the number of authorization policies attached with a service/resource does not adversely affect the performance of the overall framework.

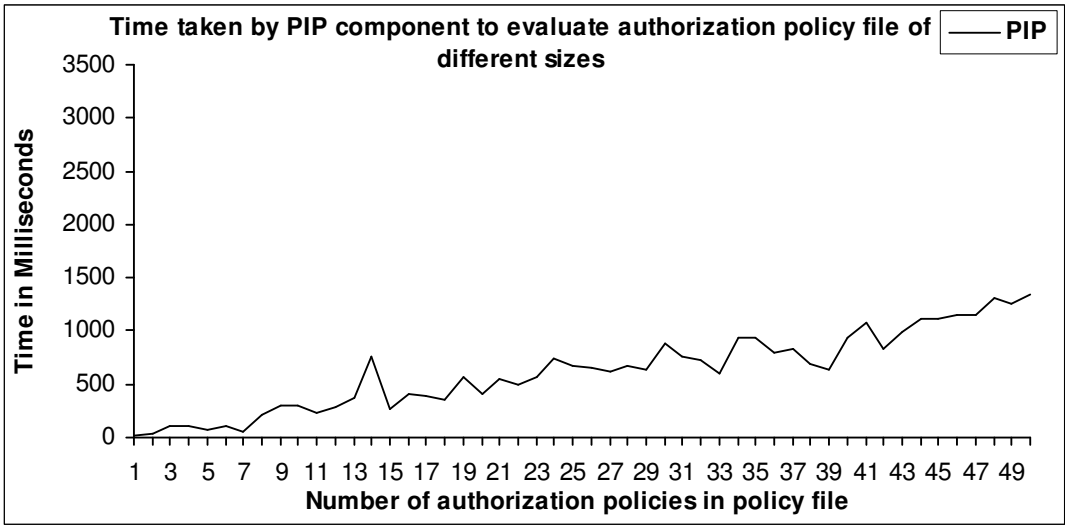

Figure 6.32: PIP time to evaluate authorization policy file of different sizes

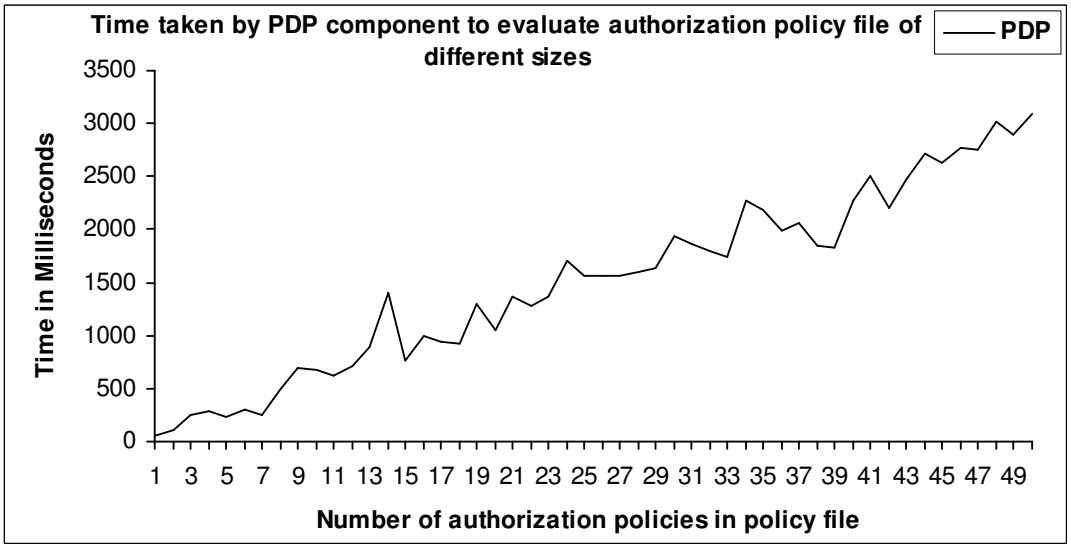

Figure 6.33: PDP time to evaluate authorization policy file of different sizes

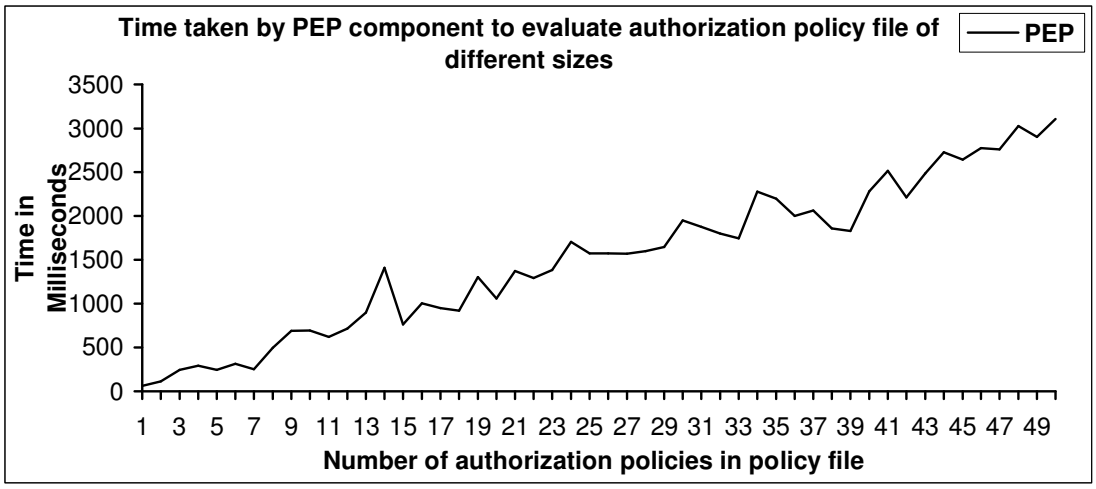

Figure 6.34: PEP time to evaluate authorization policy file of different sizes

6.2.5 Comparison of authentication, privacy, trust and authorization policies

Graphs of Figure 6.35 and Figure 6.36 show the comparison of time taken by PEP component for evaluation of authentication, privacy, trust and authorization policies individually and accumulatively. Graph of Figure 6.35 compares the time taken by PEP component to evaluate individual policy of each type and Figure 6.36 shows the effect of increase in number of policies on time taken by PEP component for all types of policies. Table 6.1 lists the average time taken by PIP, PDP and PEP components to evaluate authentication, privacy, trust and authorization policies.

Time / Policies	Average PIP Time (ms)	Average PDP Time (ms)	Average PEP Time (ms)
Authentication policies	0.3205	46.2665	53.3767
Privacy policies	24.6554	73.5858	81.3369
Trust policies	110.7392	170.0245	178.1562
Authorization policies	22.5524	76.3898	83.5802

Table 6.1: Average time taken by PIP, PEP and PDP components to evaluate authentication, privacy, trust and authorization policies.

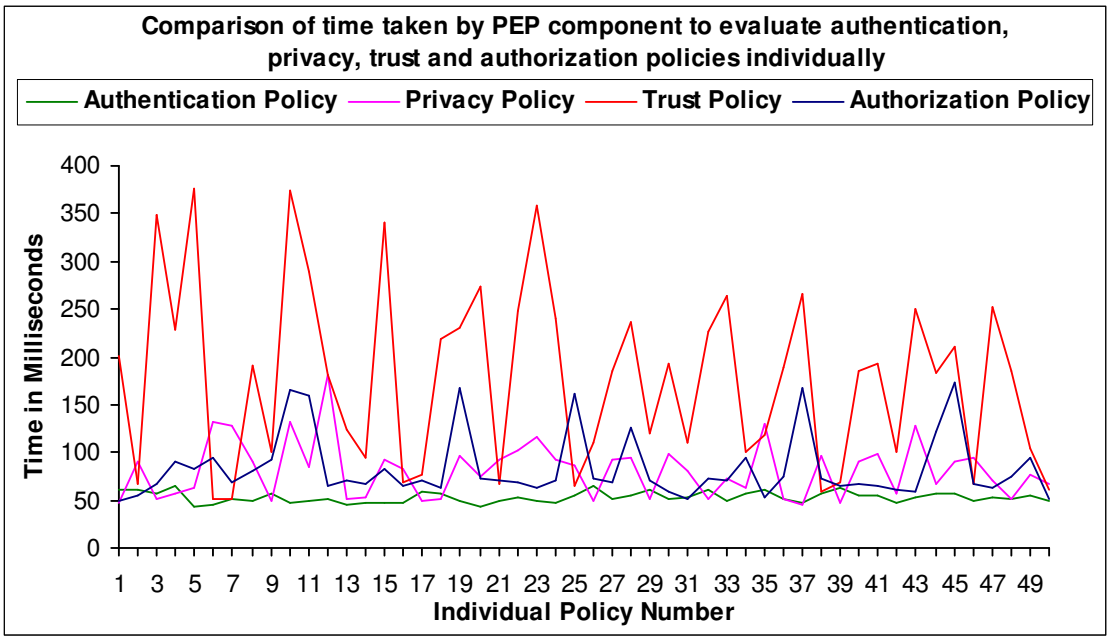

Figure 6.35: Comparison of PEP time to evaluate individual authentication, privacy, trust and authorization policies

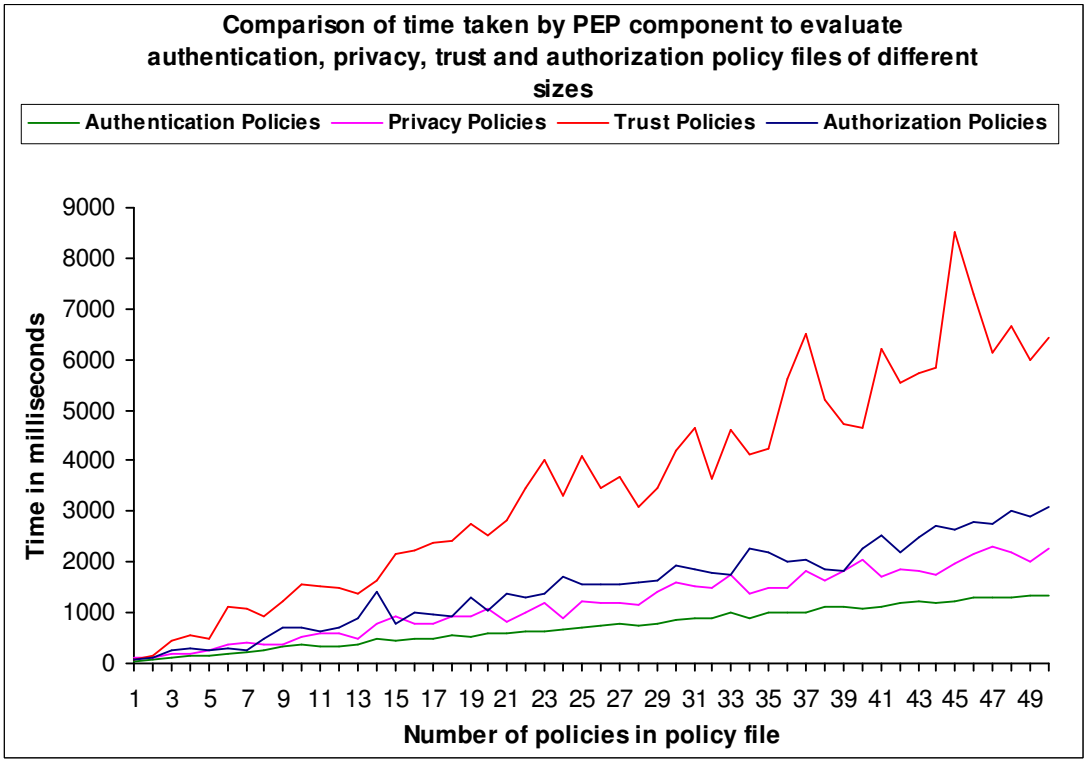

Figure 6.36: Comparison of PEP time to evaluate authentication, privacy, trust and authorization policy files of different sizes

From the graphs shown in Figure 6.35, Figure 6.36 and the values listed in Table 6.1, it is clear that authentication policies take less time and trust policies take more time to evaluate compared to evaluation of privacy and authorization policies. In trust model, the time taken by PEP component increases more assertively compared to other models. This is because of the fact that trust policies involve calculation of trust value (direct as well as recommended) which requires access to history of past interaction taken place between requester and service provider. This is a time consuming process. So trust policies take more time compared to evaluation of authentication, privacy and authorization policies, where no such calculations or access to history of past interactions is required.

Thus performance can be a concern if the number of trust policies attached with a service / resource are more. The time taken by privacy and authorization policies for their evaluation is close to each other. This is because of the fact that the attribute access and evaluation requirements of both types of policies are almost similar. Evaluation of privacy policy mainly involve checking of purpose, allowed actions and privacy conditions whose evaluation is not much different from that of authorization policies. Authentication policies take relatively less time to evaluate compared to all other types of policies because they do not require much access to resource or environment attributes. They mainly require subject attributes and authentication related information which is generally available from the access request itself. The resource and environment attribute access requirements of authentication policies are minimal. Thus they take less time to evaluate compared to all other types of policies.

Chapter 7

Conclusion and Future Scope

7.1 Conclusion

The thesis addresses important security requirements mainly identified as authentication, privacy, trust and authorization and implements a security policy framework. The security policy framework has four models namely authentication model, privacy model, trust model and policy based authorization model. These models address security requirements and policies specific to authentication, privacy, trust and authorization. The policy specification, evaluation and enforcement related functionality of authentication, privacy and trust models have been incorporated into policy based authorization model and the resulting model is called the integrated policy based authorization model. The integrated model is capable of providing access to grid services based on conformance to different types of security policies and also allow us to treat the policy involving different combinations of authentication, privacy, trust and authorization related aspects as a single unit.

To achieve the set objectives, a comprehensive review of development related to grid and web services has been done. A thorough study of standards and specifications used in grid and web services based systems is also carried out. Work done in the areas of authentication, privacy, trust and authorization in grid systems has been studied and reported in detail. Following paragraphs describe the characteristics and important features provided by individual authentication, privacy, trust and integrated policy based authorization model.

The authentication model provides support for single sign-on and delegation features using proxy certificates and a credential management service to store, retrieve and update multiple user credentials. The features provided by the authentication model are:

- ✓ It provides credential repository and a service to manage multiple user credentials.
- ✓ It is based on Web Services Security specifications like WS-Security, WS-SecureConversation etc.
- ✓ It supports the storage of multiple types of tokens/credentials. (e.g. X.509, Kerberos, Custom Security Token etc.)
- ✓ It can be used for grid as well as web services.
- ✓ CMS is distributed over different domains, therefore removes the drawbacks of central storage.
- ✓ It provides support for single sign-on and delegation through proxy certificates.
- ✓ It provides a mechanism to express, evaluate and enforce authentication policies.

The privacy model provides support for anonymous access, hidden service access and access to private information based on conformance to privacy policies. Following features have been provided by the privacy model:

- ✓ It supports purpose based access to private information/resources.
- ✓ It supports hidden service access through custom security tokens.
- ✓ It supports anonymous access through the concept of anonymous security tokens.
- ✓ It provides mechanisms to handle privacy requirements of both, the service requesters as well as service providers.
- ✓ It provides mechanisms to express, evaluate and enforce privacy policies between service providers and requesters.
- ✓ It describes the integration of privacy based access with authorization framework.
- ✓ It uses XACML to express access control privacy policies which is OASIS standard.

The trust model provides support for calculating direct as well as recommended trust to determine trustworthiness of target service to enable trust based access. Following features have been provided by the trust model:

- ✓ It can deal with identity as well as behavior trust.
- ✓ It provides support to calculate direct as well as recommended trust.
- ✓ It provides mechanisms to express, evaluate, and enforce trust policies.
- ✓ It describes the integration of trust based access with authorization framework.

- ✓ It also supports updation and management of trust relationships.

Integrated policy based authorization framework provides policy based access to grid services and integrates privacy and trust models by implementing trust and privacy based access. It provides support to express, evaluate and enforce different types of security policies (authentication, privacy, trust and authorization policies). It enables us to treat and specify the policy involving combination of authentication, privacy, trust and authorization related aspects as a single unit. The framework is flexible, scalable, supports fine grained access to services/resources and is able to express and enforce VO wide and other access control security policies. Following are the features provided by the authorization model:

- ✓ It supports fine grained and context based access to grid services/resources.
- ✓ Policy expression is platform independent.
- ✓ It is able to express and enforce VO wide and service wide access control policies.
- ✓ It introduces Filter components which make it flexible and scalable.
- ✓ It extends basic authorization mechanism to include trust and privacy based access to grid services.
- ✓ It supports multiple security policies and provides facilities to integrate different authorization mechanisms.

The security policy framework has been evaluated by implementing various security related scenarios and through implementations that involve enforcement of different types of access control policies. The implementation has been done in .NET environment with the support of WSE 3.0 toolkit.

From scenario implementations it is clear that the framework can be used to implement single sign-on and delegation features in grid systems. It can also be used to address privacy and trust related issues among service providers and service requesters. It provides support for purpose based access and anonymous access to grid services/resources. It can also be used to determine direct and recommended trust on a target service/resource. The framework is also capable of implementing policy based access to grid services and supports distributed authorization and federation of security services.

From experimental results, it is observed that total time required to evaluate different policies increases linearly with the increase in number of policies and does not become an overhead for the authorization framework. Increasing the number of policies attached with a service does not adversely affect the performance of the authorization framework. Authentication policies takes less time and trust policies takes more time to evaluate compared to evaluation of privacy and authorization policies. The performance can be a concern if the number of trust policies attached with a service / resource are more. For trust policies, the total evaluation time increases more assertively compared to other policies. This is because determination of trust value of target involves calculations and access to history of past interactions taken place between source and target which are time consuming. The time taken by privacy and authorization models in evaluating policies is close to each other. This is because both involve almost similar type of access to subject, resource, environment and other attributes. Authentication policies take less time to execute as they require little access to resource, environment and other attributes compared to other policies and almost all the information required to evaluate authentication policy is available in the access request itself.

The implementations carried out and results obtained demonstrate that the identified objectives have been achieved and the approach is workable. The framework can be used to address key security requirements related to authentication, privacy, trust and authorization, and to provide policy based access to grid services.

7.2 Future Scope

For the implementation of the framework, the necessary functionality has been developed and used wherever required, no XACML and SAML specifications implementation tools have been used. The certificates and other credentials used in the implementation have been generated through tools available in .NET environment. In future, the framework can be implemented using third party tools that provide specific implementation of many technologies and specifications. This will make the implementation and management of the framework more flexible.

The authentication model can be extended to include more types of security tokens. Work in the area of securing CMS can be started. CMS can be extended to include support for credential expiry notifications. Privacy model makes use of anonymous

security token and custom security token to provide anonymous access and hidden service access. Work on standardizing these security tokens can be started. Trust model can be extended to incorporate other attributes like honesty, accuracy, risk, time *etc*. Policy based authorization model can be extended by implementing more types of authorization mechanisms and integrating them. Work in the area of identifying and resolving conflict policies can also be started. Currently the framework provides support to express, evaluate and enforce access control policies but in future can be extended to include non access control policies like resource usage and business policies. Work can be initiated on defining and implementing an accounting and audit model. Study can be started on anti virus, intrusion detection and intrusion protection mechanisms also. Thesis has not touched the attacks that can be possible on the implemented security policy framework. A detailed analysis of different kinds of attacks and their protection mechanisms can be taken as a future work. As security is a multi dimensional problem, a lot many areas for future work exist and can be carried out.

References

[1] J. M. Crichlow, "An Introduction to Distributed and Parallel Computing", Second Edition, Prentice-Hall of India Private Limited, 2003.

[2] I. Foster, C. Kesselman and S. Tuecke, "The Anatomy of Grid: Enabling Scalable Virtual Organizations", International Journal of Supercomputer Applications, Vol. 15, No. 3, 2001.

[3] I. Foster, C. Kesselman, J.M. Nick and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", Open Grid Service Infrastructure WG, Global Grid Forum, 2002.

[4] I. Foster and C. Kesselman (eds), "The Grid: Blueprint for a New Computing Infrastructure", Morgan Kaufmann, 1999.

[5] B. Jacob, M. Brown, K. Fukui and N. Trivedi, "Introduction to Grid Computing", IBM Redbooks, 2005, available at http://www.redbooks.ibm.com/abstracts/sg246778.html

[6] J. Joseph and C. Fellenstein, "Grid Computing", Pearson Education, 2004.

[7] M. Parashar and C. A. Lee, "Grid Computing: Introduction and Overview", IEEE Special Issue on Grid Computing, Vol. 93, No. 3, March 2005, available at http://www.caip.rutgers.edu/TASSL/Papers/proc-ieee-intro-04.pdf

[8] C. Comito, D. Talia and P. Trunfio, "Grid Services: principles, implementations and use", International Journal of Web and Grid Services, Vol. 1, No. 1, 2005, pp. 48-68.

[9] R. Buyya, "Economic-based Distributed Resource Management and Scheduling for Grid Computing", Ph.D. Thesis, Monash University, Melbourne, Australia, April 12, 2002.

[10] J. Nabrzyski, J. M. Schopf and J. Weglarg, "Grid Resource Management State of the Art and Future Trends", Kluwer Academic Publishers, 2004.

[11] S. Bawa and G. K. Sharma, "A Parallel Transitive Closure Computation Algorithm for VLSI Test Generation", Lecture Notes in Computer Science, Vol. 2367, 2002.

[12] P. Chandrapal and P. Kumar, "A Scalable Multi-level Distributed System-Level Diagnosis", International Conference on Distributed Computing and Internet Technology (ICDCIT), Bhubaneswar, India, December 22-24, 2005.

[13] G. Agha and C. A. Varela, "Worldwide Computing Middleware," Practical Handbook on Internet Computing, Editor: M. Singh, Invited Chapter, CRC Press, 2004.

[14] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris and D. Orchard, “Web Services Architecture”, W3C Working Group, 2004, available at http://www.w3.org/TR/ws-arch

[15] http://www.w3.org

[16] http://www.ggf.org

[17] I. Foster, C. Kesselman, J.M. Nick and S. Tuecke, “Grid Services for Distributed System Integration”, IEEE Computer, Vol. 35, Issue 6, June 2002, pp. 37-46.

[18] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling and P. Vanderbilt, ”Open Grid Services Infrastructure (OGSI) Version 1.0”, OGSI WG, June 2003, available at http:// www.globus.org/toolkit/draft-ggf-ogsi-gridservice-33_2003-06-27.pdf

[19] http://www.globus.org/wsrf

[20] http://www.oasis-open.org

[21] Borja Sotomayor, “The Globus Toolkit 4 Programmer’s Tutorial”, 2005, available at http://gdp.globus.org/gt4-tutorial

[22] Borja Sotomayor, “The Globus Toolkit 3 Programmer’s Tutorial”, 2004, available at http://gdp.globus.org/gt3-tutorial

[23] L. Ramakrishnan, “Securing Next Generation Grids”, IEEE IT Pro, Volume 6, Issue 2, March-April 2004, pp. 34-39.

[24] I. Foster, C. Kesselman, G. Tsudik and S. Tuecke, “A Security Architecture for Computational Grids”, 5th ACM Conference on Computer and Communications Security, California, USA, 1998, pp. 83-92.

[25] S. Singh and S. Bawa, “A Framework for Handling Security Problems in Grid Environment using Web Services Security Specifications”, Second International Conference on Semantics, Knowledge and Grid (SKG2006), Guilin, China, November 2006, pp. 68.

[26] S. Singh and S. Bawa, “Security Policies: Key Factor for Success of Grid Services”, International Conference on Challenges and Opportunities in IT Industry (ICCII-2005), Ludhiana, India, November 2005.

[27] http://www.w3.org/xml

[28] S. Grahm, D. Davis, S. Simeonov, G. Daniels, P. Brittenham, Y. Nakamura, P. Fremantle, D. Konig and C. Zentner, "Building Web Services with Java", Second Edition, Pearson Education, 2005.

[29] http://www.w3.org/TR/soap

[30] R. Basiura, M. M. Fecchio, M. Batongbacal, B. Bohling, M. Clark, A. Eide, R. Eisenberg, B. Loesgen, C. L. Miller, M. Reynolds, B. Sempf and S. Sivakumar, "Professional ASP .NET Web Services", Apress, 2003.

[31] http://www.w3.org/TR/wsdl

[32] http://uddi.xml.org

[33] "Security in a Web Services World: A Proposed Architecture and Roadmap", Joint Security whitepaper from IBM Corporation and Microsoft Corporation, 2002, available at http://www.ibm.com/developerworks/library/specification/ws-secmap

[34] B. Atkinson, G. Della-Libera, S. Hada, M. Hondo, P. Hallam-Baker, C. Kaler, J. Klein, B. LaMacchia, P. Leach, J. Manferdelli, H. Maruyama, A. Nadalin, N. Nagaratnam, H. Prafullchandra, J. Shewchuk and D. Simon, "Web Services Security (WS-Security)", 2002, available at http://www.verisign.com/wss/wss.pdf

[35] S. Anderson, J. Bohren, T. Boubez, M. Chanliau, G. Della-Libera, B. Dixon, P. Garg, M. Gudgin, P. Hallam-Baker, M. Hondo, C. Kaler, H. Lockhart, R. Martherus, H. Maruyama, A. Nadalin, N. Nagaratnam, A. Nash, R.Philpott, D. Platt, H. Prafullchandra, M. Sahu, J. Shewchuk, D. Simon, D. Srinivas, E. Waingold, D. Waite, D. Walter and R. Zolfonoon, "Web Services Trust Language(WS-Trust)", 2005, available at http://specs.xmlsoap.org/ws/2005/02/trust/WS-Trust.pdf

[36] S. Bajaj, D. Box, D. Chappel, F. Curbera, G. Daniels, P. Hallm-Baker, M. Hondo, C. Kaler, D. Langworthy, A. Nadalin, N. Nagaratnam, H. Prafullchandra, C. von Riegen, D. Roth, J. Schlimmer, C. Sharp, J. Shewchuk, A. Vedamuthu, U. Yalcinalp and D. Orchard, "Web Services Policy Framework (WS-Policy)", 2006, available at http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-polfram/ws-policy-2006-03-01.pdf

[37] S. Anderson, J. Bohren, T. Boubez, M. Chanliau, G. Della-Libera, B. Dixon, P. Garg, M. Gudgin, S. Hada, P. Hallam-Baker, M. Hondo, C. Kaler, H. Lockhart, R. Martherus, H. Maruyama, A. Nadalin, N. Nagaratnam, A. Nash, R.Philpott, D. Platt, H. Prafullchandra, M. Sahu, J. Shewchuk, D. Simon, D. Srinivas, E.

Waingold, D. Waite, D. Walter and R. Zolfonoon, "Web Services Secure Conversation Language(WS-SecureConversation)", 2005, available at, http://specs.xmlsoap.org/ws/2005/02/sc/WS-SecureConversation.pdf

[38] H. Lockhart, S. Andersen, J. Bohren, Y. Sverdlov, M. Hondo, H. Maruyama, A. Nadalin, N. Nagaratnam, T. Boubez, K. S. Morrison, C. Kaler, A. Nanda, D. Schmidt, D. Walters, H. Wilson, L. Burch, D. Earl, S. Bajaj and H. Prafullchandra, "Web Services Federation Language (WS-Federation)", 2006, available at http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-fed/WS-Federation-V1-1B.pdf

[39] http://xml.coverpages.org/saml.html

[40] M. O'Neill, P. Hallam-Baker, S. M. Cann, M. Shema, E. Simon, P. A. Watters and A.White, "Web Services Security", Tata McGraw-Hill Publishing Company Limited, 2003.

[41] XACML Version 2.0, 2005, available at http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf

[42] M. Verma, "Control information access with XACML", 2004, article available at http://www.ibm.com/developerworks/xml/library/x-xacml

[43] N. Nagaratnam, P. Janson, J. Dayka, A. Nadalin, F. Siebenlist, V. Welch, I. Foster and S. Tuecke, "Security Architecture for Open Grid Services", GGF OGSA Security Workgroup, 2003.

[44] H. Casanova, "Distributed Computing Research Issues in Grid Computing", ACM SIGACT News Distributed Computing Column 8, Volume 33, Issue 3, 2002, pp. 50-70.

[45] P. J. Broadfoot and A. P. Martin, "A Critical Survey of Grid Security Requirements and Technologies", Programming Research Group, Oxford University Computing Laboratory, PRG-RR-03-15, 2003.

[46] M. Humphrey and M. Thompson, "Security Implications of Typical Grid Computing Usage Scenarios," 10th IEEE International Symposium on High Performance Distributed Computing (HPDC), San Francisco, CA, USA, August 2001, pp. 95-103.

[47] Marty Humphrey, Mary R. Thompson and Keith R. Jackson, "Security for Grids", Invited Paper, Proceedings of the IEEE Special Issue on Grid Computing, Vol. 93, No. 3, March 2005, pp. 644-652.

[48] Anirban Chakrabarti, "Grid Computing Security", Springer, 2007.

[49] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman and S. Tuecke, "Security for Grid Services", 12th International Symposium on High Performance Distributed Computing, Seattle, WA, USA, 2003, pp. 48-57.

[50] F. Siebenlist, V. Welch, S. Tuecke, I. Foster, N. Nagaratnam, P. Janson J. Dayka and A. Nadalin, "OGSA Security Roadmap", 2002, available at http://www.cs.virginia.edu/~humphrey/ogsa-sec-wg/ogsa-sec-roadmap-v13.pdf

[51] http://www.legion.virginia.edu/security_arch.html

[52] E. Belani, A. Vahdat, T. Anderson and M. Dahlin, "The CRISIS Wide Area Security Architecture", Seventh USENIX Security Symposium, Texas, 1998, pp. 15-30.

[53] "Global Security Architecture", EU Deliverable DJRA3.1, 2004, available at https://edms.cern.ch/document/487004

[54] D. Jana, A. Chaudhuri, A. Datta and B. B. Bhaumik, "Framework for Handling Security Issues in Interoperable Grid Services", IEEE Indicon Conference, Chennai, India, December 2005, pp. 280-285.

[55] Q. Zhou, G. Yang, J. Shen and C. Rong, "A Scalable Security Architecture for Grid", Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies, Dalian, China, 2005, pp. 89-93.

[56] Y. Demchenko, L. Gommans, C. de Laat, B. Oudenaarde, A. Tokmakoff, M. Snijders and R. van Buuren, "Security Architecture for Open Collaborative Environment", Book Chapter, Advances in Grid Computing – EGC 2005, Lecture Notes in Computer Science, Vol. 3470, 2005, pp. 589-599.

[57] X. WU, G. Yang, J. Shen and Q. Zhou, "A Novel Security Model based on Virtual Organization for Grid", Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'05), Dalian, China, December 2005, pp. 106-109.

[58] D. Agarwal, M. Lorch, M. Thompson and M. Perry, "A New Security Model for Collaborative Environment", Lawrence Berkeley National Laboratory, Paper LBNL-52894, 2003, available at http://repositories.cdlib.org/lbnl/LBNL-52894

[59] S. Naqvi and M. Riguidel, "Security Architecture for Heterogeneous Distributed Computing Systems", 38th Annual International Carnahan Conference on Security Technology, Albuquerque, New Mexico, 2004, pp. 34-41.

[60] C. de Laat, G. Gross, L. Gommans, J. Vollbrecht and D. Spence, "Generic AAA Architecture", RFC 2903, 2000, available at http://www.ietf.org/rfc/rfc2903.txt

[61] K. Keahey and V. Welch, "Fine-Grain Authorization for Resource Management in the Grid Environment", Third International Workshop on Grid Computing, Baltimore, MD, USA, 2002, pp. 199-206.

[62] K. A. Usmani and N. Prakash, "Information Security Issues in Wireless Networks", CSI Communications, Vol. 31, No. 5, August, 2007.

[63] V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder and F. Siebenlist, "X.509 Proxy Certificates for Dynamic Delegation", Third Annual PKI R&D Workshop, Gaithersburg, MD, USA, 2004, available at globus.org/alliance/publications/papers/pki04-welch-proxy-cert-final.pdf

[64] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson, "Internet X.509 Public Key Infrastructure (PKI) proxy certificate profile," IETF RFC 3820, June 2004, available at http://www.ietf.org/rfc/rfc3820.txt

[65] "Microsoft Expands Passport to Enable Universal Single Sign-In", Microsoft Press Release, 20 September 2001, available at http://www.microsoft.com/ presspass/press/2001/sep01/09-20PassportFederationPR.asp

[66] "Entrust TruePass Product Portfolio: Technical Overview", July 2003, available at http://www.entrust.com/resources/download.cfm/21136#search="TruePass portfolio"

[67] J. Novotny, S. Tuecke and V. Welch, "An Online Credential Repository for the Grid: MyProxy", Tenth International Symposium on High Performance Distributed Computing, San Francisco, CA, USA, 2001, pp. 104-111.

[68] "KX.509/KCA", 2002, available at http://archive.nsf-middleware.org/documentation/ NMI-R5/0/gridscenter/KX509KCA

[69] B. Schneier, "Password Safe: The Security of Blowfish in a Password Database", 2003, available at http://www.schneier.com/passsafe.html

[70] "Microsoft Windows 'TrustBridge' to enable organizations to share user identities across business boundaries", 2002, available at http://www.microsoft.com/presspass/press/2002/Jun02/06-06TrustbridgePR.mspx

[71] J. Kemp, Ed. "Liberty ID-WSF", May 2004, available at https://www.projectliberty.org/resources/whitepapers/Liberty_ID-WSF_Web_Services_ Framework.pdf

[72] D. D. Vecchio, M. Humphrey, J. Basney and N. Nagaratnam, "CredEx: User Centric Credential Management for Grid and Web Services", IEEE International Conference on Web Services, Orlando, Florida, USA, 2005, pp. 149-156.

[73] R. Sandhu, M. Bellare and R. Ganesan, "Password-Enabled PKI: Virtual Smart Cards versus Virtual Soft Tokens", First Annual PKI Research Workshop, Gaithersburg, MD, USA, 2002, pp. 89-96.

[74] V. Welch, K. Keahey, F. Siebenlist and T. Barton, "NMI development: Policy Controlled Attribute Framework – A Privacy Friendly Framework for Policy Enforced Release, Sharing, and Matching of Attribute Information", Project Summary, National Science Foundatuion, National Middleware Infrastructure Solicitation NSF 04-555, 2004.

[75] "Globus Toolkit", available at http://www.globus.org/toolkit

[76] http://www.globus.org/security/overview.html

[77] The Globus Security Team, "Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective", 2005, available at http://www.globus.org/toolkit/docs/4.0/security/GT4-GSI-Overview.pdf

[78] D. Jana and B. B. Bhaumik, "Single Signon for Grid Services", IEEE INDICON First Indian Annual Conference, Kharagpur, India, December 2004, pp. 513-516.

[79] D. Kouril and J. Basney, "A Credential Renewal Service for Long-Running Jobs", Sixth IEEE International Workshop on Grid Computing, Seattle, USA, 2005, pp. 63-68.

[80] M. Lorch, J. Basney and D. Kafura, "A Hardware-secured Credentials Repository for Grid PKIs", 4th IEEE/ACM International Symposium on Cluster Computing and the Grid, Chicago, Illinois, USA, 2004, pp. 640-647.

[81] R. Butler, V. Welch, D. Engert, I. Foster, S. Tuecke, J. Volmer and C. Kesselman, "A National-Scale Authentication Infrastructure", IEEE Computer, Vol. 33, Issue 12, December 2000, pp. 60-66.

[82] H. R. Nagesh, K. Chandra Sekaran and K. M. Hebbar, "Design, implementation and performance analysis of secure group communication", 15th International Conference on Computer Communication, Mumbai, India, 2002, pp. 880-891.

[83] G. Karjoth and M. Schunter, "A Privacy Policy Model for Enterprises", 15th IEEE Computer Security Foundations Workshop, Nova Scotia, Canada, June 24-26, 2002, pp. 271.

[84] E. Damiani, S. De Capitani di Vimercati and P. Samarati, "Privacy Enhanced Authorizations and Data Handling", W3C Standardization Workshop, Ispra, 2006, available at http://www.w3.org/2006/07/privacy-ws/papers/31-samarati-privacy-enhanced-authorizations

[85] S. Singh and S. Bawa, "A Privacy Policy Framework for Grid and Web Services", Information Technology Journal, Vol. 6, No. 6, 2007, pp. 809-817.

[86] A. Brar and J. Kay, "Privacy and Security in Ubiquitous Personalized Applications", 2005, available at http://www.isr.uci.edu/pep05/papers/PEPp.pdf

[87] S. Fischer-Hubner and A. Ott, "From a Formal Privacy Model to its Implementation", 21st National Information Systems Security Conference, 1998, available at http://www.cs.kau.se/~simone/niss98.pdf

[88] Shibboleth Architecture, 2005, available at http://shibboleth.internet2.edu/docs/internet2-mace-shibboleth-arch-protocols-200509.pdf

[89] M. Lorch, D.B. Adams, D. Kafura, M.S.R. Koneni, A. Rathi and S. Shah, "The PRIMA System for Privilege Management, Authorization and Enforcement in Grid Environments", Fourth International Workshop on Grid Computing, Phoenix, Arizona, 2003, pp. 109.

[90] D. W. Chadwick, A. Otenko and E. Ball, "Implementing Role Based Access Controls Using X.509 Attribute certificates - the PERMIS Privilege Management Infrastructure", 2002, available at http://sec.isi.salford.ac.uk/download/InternetComputingPaperv4.pdf

[91] Platform for Privacy Preferences (P3P) Project, http://www.w3.org/P3P

[92] P. Ashley, S. Hada, G. Karjoth, and M. Schunter, "E-P3P privacy policies and privacy authorization", ACM workshop on Privacy in the Electronic Society (WPES 2002), Washington, DC, USA, November 2002, pp. 103-109.

[93] A. Luther, R. Buyya, R. Ranjan and S. Venugopal, "Alchemi: A .NET-based Grid Computing Framework and its Integration into Global Grids", Technical Report, GRIDS-TR-2003-8, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, December 2003, available at http://www.gridbus.org/~alchemi/files/alchemi_techreport.pdf

[94] L. Cranor, M. Langheinrich and Massimo Marchiori, "A P3P Preference Exchange Language 1.0 (APPEL 1.0)", 2002, available at http://www.w3.org/TR/2002/WD-P3P-preferences-20020415

[95] M. Langheinrich, "A Privacy Awareness System for Ubiquitous Computing Environments", Lecture Notes in Computer Science, Vol. 2498, 2002, pp. 237-245.

[96] P. Bonatti, E. Damiani, S. De Capitani di Vimercati and P. Samarati, "An access control system for data archives", 16th International Conference on Information Security, Paris, France, 2001.

[97] S. Lederer, A. K. Dey and J. Mankoff, "Everyday Privacy in Ubiquitous Computing Environments", 2002, available at guir.berkeley.edu/privacyworkshop2002/papers/lederer-ubicomp02-workshop.pdf

[98] S. Crane, "Privacy Preserving Trust Agents", HP Technical Report, HPL-2004-197, 2004, available at http://www.hpl.hp.com/techreports/2004/HPL-2004-197.pdf

[99] U.M. Mbanaso, G.S. Cooper, D.W. Chadwick and S. Proctor, "Privacy Preserving Trust Authorization Framework Using XACML", International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'06), Buffalo, NY, USA, 2006, pp. 673-678.

[100] W. Xu, V.N. Venkatakrishanan, R. Sekar and I.V. Ramakrishanan, "A Framework for Building Privacy-Conscious Composite Web Services", IEEE International Conference on Web Services, Chicago, USA, September 2006, pp. 655-662.

[101] C. Schlager, T. Nowey and J. A. Montenegro, "A Reference Model for Authentication and Authorization Infrastructures Respecting Privacy and Flexibility in b2c eCommerce", First International Conference on Availability, Reliability and Security (ARES'06), 2006.

[102] D. Olmedilla, O. F. Rana, B. Matthews and W. Nejdl, "Security and Trust Issues in Semantic Grids", 2005, available at http://drops.dagstuhl.de/opus/volltexte/2006/408

[103] C. Lin, V. Varadharajan, Y. Wang and V. Pruthi, "Enhancing Grid Security with Trust Management", IEEE International Conference on Service Computing, Shanghai, China, 2004, pp. 303-310.

[104] D. Gambetta, "Trust: Making or Breaking of Cooperative Relations", Blackwell Publishers, 1990.

[105] T. Grandison and M. Sloman, "A Survey of Trust in Internet Applications", IEEE Communications Surveys and Tutorials, Vol. 3, Issue 4, 2000, pp. 2-16.

[106] F. Azzedin and M. Maheswaran, "Evolving and Managing Trust in Grid Computing Systems", IEEE Canadian Conference on Electrical and Computer Engineering, Manitoba, Canada, 2002, pp. 1424-1429.

[107] A. Abdul-Rahman and S. Hailes, "Supporting Trust in Virtual Communities", 33rd Annual Hawaii International Conference on System Sciences, Maui, Hawaii, 2000, pp. 6007-6015.

[108] Q. Zhang, T. Yu and K. Irwin, "A Classification Scheme for Trust Functions in Reputation-Based Trust Management", Workshop on Trust, Security, and Reputation on the Semantic Web", Hiroshima, Japan, 2004.

[109] A. Abdul-Rahman and S. Hailes, "A Distributed Trust Model", Workshop on New Security Paradigms, Virginia, USA, 1997, pp. 48-60.

[110] M. Blaze, "Using the KeyNote Trust management System", 1999, article available at http://www.crypto.com/trustmgt/kn.html

[111] M. Blaze, J. Feigenbaum and J. Lacy, "Decentralized Trust Management", IEEE Symposium on Security and Privacy, Oakland, CA, USA, 1996, pp. 164-173.

[112] M. R. Thompson, D. Olson, R. Cowles, S. Mullen and M. Helm, "CA-based Trust Model for Grid Authentication and Identity Delegation", Grid Certificate Policy WG, 2002, available at http://www.gridcp.es.net/Documents/GGF6/TrustModel-final.pdf

[113] L. Xiong and L. Liu, "PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities", IEEE Transactions of Knowledge and Data Engineering, Vol. 16, No. 7, pp. 179-194, 2004.

[114] B. Dragovic and E. Kotsovinos, "XenoTrust: Event-based distributed trust management", Second International Workshop on Trust and Privacy in Digital Business, Prague (Czech Republic), 2003, pp. 410-414.

[115] S. Lee, R. Sherwood and B. Bhattacharjee, "Cooperative peer groups in NICE", IEEE INFOCOM, San Francisco (CA), USA, 2003, pp. 1272-1282.

[116] S. Song, K. Hwang and M. Macwan, "Fuzzy Trust Integration for Security Enforcement in Grid Computing", International Conference on Network and Parallel Computing (NPC), Wuhan (China), 2004, pp. 9-21.

[117] S. Song, K. Hwang and Y. K. Kwok, "Trusted Grid Computing with Security Binding and Trust Integration", Journal of Grid Computing, Vol. 3, No. 1, 2005, pp. 24-34.

[118] M. Winslett, T. Yu, K. E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith and L. Yu, "Negotiating Trust on the Web", IEEE Internet Computing, Vol. 7, No. 6, pp. 45-52, 2002.

[119] H. Lei, G. C. Shoja, "A Distributed Trust Model for e-Commerce Applications", IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05), Hong Kong, 2005, pp. 290-293.

[120] R. Chen and W. Yeager, "Poblano A Distributed Trust Model for Peer-to-Peer Networks", Technical Report, Sun Microsystems, 2001, pp. 1-26, available at http://gnunet.org/papers/jxtatrust.pdf

[121] F. Azzedin, M. Maheswaran and A. Mitra, "Trust Brokering and Its Use for Resource Matchmaking in Public-Resource Grids", Journal of Grid Computing, Vol. 4, No. 3, 2006, pp. 247-263.

[122] F. Azzedin and M. Maheswaran, "Integrating Trust into Grid Resource Management Systems", International Conference on Parallel Processing (ICPP), Canada, 2002, pp. 47-54.

[123] L. Pearlman, V. Welch, I. Foster, C. Kesselman and S. Tuecke, "A Community Authorization Service for Group Collaboration", IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, CA, USA, 2002, pp. 50-59.

[124] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell' Agnello, A. Frohner, A. Gianoli, K. Lorentey and F. Spataro, "VOMS, an Authorization System for Virtual Organizations", Lecture Notes in Computer Science, Vol. 2970, 2004, pp. 33-40.

[125] K. Stoupa and A. Vakali, "Policies for Web Security Services", 2005, available at http://citeseer.ist.psu.edu/stoupa05policies.html

[126] B., I. Foster, F. Siebenlist, R. Ananthakrishnan and T. Freeman, "A Multipolicy Authorization Framework for Grid Security", Fifth IEEE International Symposium on Network Computing and Applications, USA, 2006, pp. 269-272.

[127] S. Singh and S. Bawa, "A Privacy, Trust and Policy based Authorization Framework for Services in Distributed Environments", International Journal of Computer Science, Vol. 2, No. 2, 2007, pp. 85-92.

[128] K. Keahey, V. Welch, S. Lang, B. Liu and S. Meder, "Fine-Grain Authorization Policies in the GRID: Design and Implementation", International Workshop on Middleware for Grid Computing, Canada, 2003.

[129] M. Lorch, B. Cowles, R. Baker, L. Gommans, P. Madsen, A. McNab, L. Ramakrishnan, K. Sankar, D. Skow and M. R. Thompson, "Conceptual Grid Authorization Framework and Classification", Authorization Frameworks and Mechanisms-WG, 2003, available at http://www.ogf.org/documents/GFD.38.pdf

[130] M. Thompson, A. Essiari and S. Mudumbai, "Certificate-based Authorization Policy in a PKI Environment", ACM Transactions on Information and System Security, Vol. 6, Issue 4, 2003, pp. 566-588.

[131] R. Lepro, "Cardea: Dynamic Access Control in Distributed Systems", NAS Technical Report NAS-03-020, November 2003.

[132] V. Welch, T. Barton, K. Keahey and F. Siebenlist, "Attributes, Anonymity, and Access: Shibboleth and Globus Integration to Facilitate Grid Collaboration", 4th annual PKI R&D Workshop, USA, April 2005, available at http://grid.ncsa.uiuc.edu/papers/gridshib-pki05-final.pdf

[133] http://gridshib.globus.org/about.html

[134] R. Baker, D. Yu and T. Wlodek, "A Model for Grid User Management", Computing in High Energy and Nuclear Physics, La Jolla, California, 24-28 March, 2003, ePrint cs.DC/0306063

[135] G. Zhang and M. Parashar, "SESAME: Scalable, Environment Sensitive Access Management Engine", Cluster Computing, Vol. 9, Issue 1, 2006, pp. 19-27.

[136] H. Jin, W. Qiang, X. Shi and D. Zou, "RB-GACA: A RBAC based grid access control archirecture", International Journal of Grid and Utility Computing, Vol. 1, No. 1, 2005, pp. 61-70.

[137] S. Cannon, S. Chan, D. Olson, C.Tull, V. Welch and L. Pearlman, "Using CAS to Manage Role-Based VO Sub-Groups", Conference Proceedings of Computing in High Energy and Nuclear Physics (CHEP03), La Jolla, 2003.

[138] http://www.ipg.nasa.gov

[139] X. Zhang, M. Nakae, M. J. Covington and R. Sandhu, "A Usage-based Authorization Framework for Collaborative Computing Systems", Eleventh ACM symposium on Access control models and technologies (SACMAT'06), California, USA, 2006, pp. 180-189.

[140] X. Zhang, M. Nakae, M. J. Covington and R. Sandhu, "Towards a Usage-Based Security Framework for Collaborative Computing Systems", ACM Transactions on Information and System Security, Vol. 11, No. 1, 2008.

[141] J. Wu, C. B. Leangsuksum, V. Rampure and H. Ong, "Policy-based Access Control Framework for Grid Computing", Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06), Singapore, 2006, pp. 391-394.

[142] L. Seitz, E. Rissanen, T. Sandholm, B. S. Firozabadi and O. Mulmo, "Policy Administration Control and Delegation using XACML and Delegent", Grid Computing Workshop, Seattle, USA, 2005, pp. 49-54.

[143] G. Wasson and M. Humphrey, "Policy and Enforcement in Virtual Organizations", Fourth International Workshop on Grid Computing (GRID'03), 2003, pp. 125.

[144] G. Wasson and M. Humphrey, "Towards Explicit Policy Management for Virtual Organizations", IEEE International Workshop on Policies for Distributed Systems and Networks, 2003, pp. 173.

[145] J. Feng, G. Wasson and M. Humphrey, "Resource Usage Policy Expression and Enforcement in Grid Computing", 8th IEEE/ACM International Conference on Grid Computing (GRID 2007), Texas, 2007, pp. 66-73.

[146] D. Marriott and M. Sloman, "Management Policy Service for Distributed Systems", IEEE Third International Workshop on Services in Distributed and Networked Environments (SDNE'96), 1996, pp. 2-9.

[147] K. Yang, A. Galis and C. Todd, "Policy-based Active Grid Management Architecture", 10th IEEE International Conference on Networks (ICON), 2002.

[148] R. Hull, B. Kumar and D. Lieuwen, "Towards Federated Policy Management", Fourth International Workshop on Policies for Distributed Systems and Networks (POLICY'03), 2003, pp. 183.

[149] R. Neisse, E. D. V. Pereira, L. Z. Granville, M. J. B. Almeida and L. M. R. Tarouco "An Hierarchical Policy-Based Architecture for Integrated Management of Grids and Networks", Fifth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'04), 2004, pp. 103.

[150] N. N. Vuong, G. S. Smith and Y. Deng, "Managing Security Policies in a Distributed Environment Using eXtensible Markup Language (XML)", ACM Symposium on Applied Computing, Nevada, USA, 2001, pp. 405-411.

[151] S. Indrakanti, V. Varadharajan and R. Agarwal, "On the design, implementation and application of an authorisation architecture for web services", International Journal of Information and Computer Security, Vol. 1, No. 1/2, 2007, pp. 64-108.

[152] C. Dumitrescu, M. Wilde and I. Foster, "Policy-based Resource Allocation for Virtual Organizations", Technical Report, 2003, available at http://www.griphyn.org/documents/document_server/uploaded_documents/doc--683--vo-policy.v16.pdf

[153] F. J. G. Clemente, G. M. Perez, O. C. Reverte and A. F. G. Skarmeta, "A Proposal of a CIM-based Policy Management Model for the OGSA Security Architecture", Workshop on Grid Computing and its Applications to Data Analysis (GADA), Lecture Notes in Computer Science, Vol. 3292, 2004, pp. 165-174.

[154] M. Lorch, D. Kafura and S. Shah, "An XACML-based Policy Management and Authorization Service for Globus Resources", Fourth International Workshop on Grid Computing (GRID'03), 2003, pp. 208-210.

[155] J. F. da Silva, L. P. Gaspary, M. P. Barcellos and A. Detsch, "Policy-based Access Control in Peer-to-Peer Grid Systems", Grid Computing Workshop, Seattle, USA, 2005, pp. 107-113.

[156] L. Bauer, M. A. Schneider and E. W. Felten, "A General and Flexible Access-Control System for the Web", 11th USENIX Security Symposium, San Francisco, USA, August, 2002, pp. 93-108.

[157] https://cabig.nci.nih.gov/

[158] http://cagrid.org/

[159] http://ngrid.sourceforge.net/

[160] J. Feng, L. Cui, G. Wasson and M. Humphrey, "Toward Seamless Grid Data Access: Design and Implementation of GridFTP on .NET", Sixth IEEE International Workshop on Grid Computing, Seattle, USA, Nov 13-14, 2005, pp. 164-171.

[161] G. Wasson, N. Beekwilder, D. Del Vecchio, M. Morgan and M. Humphrey, "Resource-Oriented Computing: Design, Implementation, and Evaluation of WSRF.NET", Journal of Grid Computing, Vol. 6, No. 2, 2008, pp. 177-194.

[162] G. Wasson, N. Beekwilder, M. Morgan and M. Humphrey, "OGSI.NET: OGSI-compliance on the .NET Framework", 4th IEEE/ACM International Symposium on Cluster Computing and the Grid (ccGrid 2004), Chicago, Illinois, April 19-22, 2004, pp. 648-655.

List of Publications

Journal Publications

1. Sarbjeet Singh and Seema Bawa, "Proxy Based Authentication and Management of Multiple User Credentials for Grid Services", International Journal of Systemics, Cybernetics and Informatics (IJSCI), April 2008, pp. 37-43.
2. Sarbjeet Singh and Seema Bawa, "A Privacy Policy Framework for Grid and Web Services", Information Technology Journal, Vol. 6, No. 6, 2007, pp. 809-817.
3. Sarbjeet Singh and Seema Bawa, "A Privacy, Trust and Policy Based Authorization Framework for Services in Distributed Environments", International Journal of Computer Science (IJCS), Vol. 2, No. 2, 2007, pp. 85-92.

International Conference Publications

4. Sarbjeet Singh and Seema Bawa, "Design of a Framework for Handling Security Issues in Grids", Ninth International Conference on Information Technology (ICIT2006), Published by IEEE Computer Society Press, Bhubaneswar, India, December 2006, pp. 178-179.
5. Sarbjeet Singh and Seema Bawa, "A Framework for Handling Security Problems in Grid Environment using Web Services Security Specifications", Second International Conference on Semantics, Knowledge and Grid (SKG2006), Published by IEEE Computer Society Press, Guilin, China, November 2006, pp. 68.
6. Sarbjeet Singh and Seema Bawa, "Security Policies: Key Factor for Success of Grid Services", International Conference on Challenges and Opportunities in IT Industry (ICCII2005), Ludhiana, India, November 2005.

National Conference Publications

7. Sarbjeet Singh, "An Integrated Approach to Security Design for Grid and Web Services", National Conference on Emerging Trends in Wireless Communication and E-Security, Chandigarh, India, January 2007, pp. 121-126.