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.

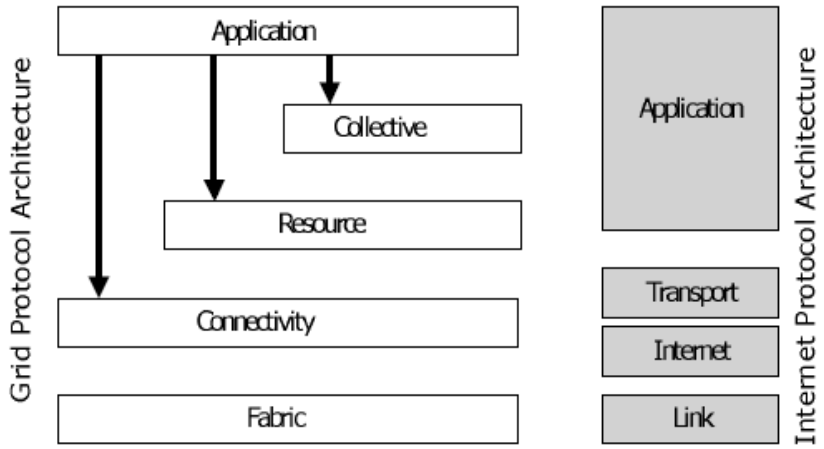

Figure 1.1: The layered Grid architecture and its relationship to the Internet protocol architecture [2]

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.

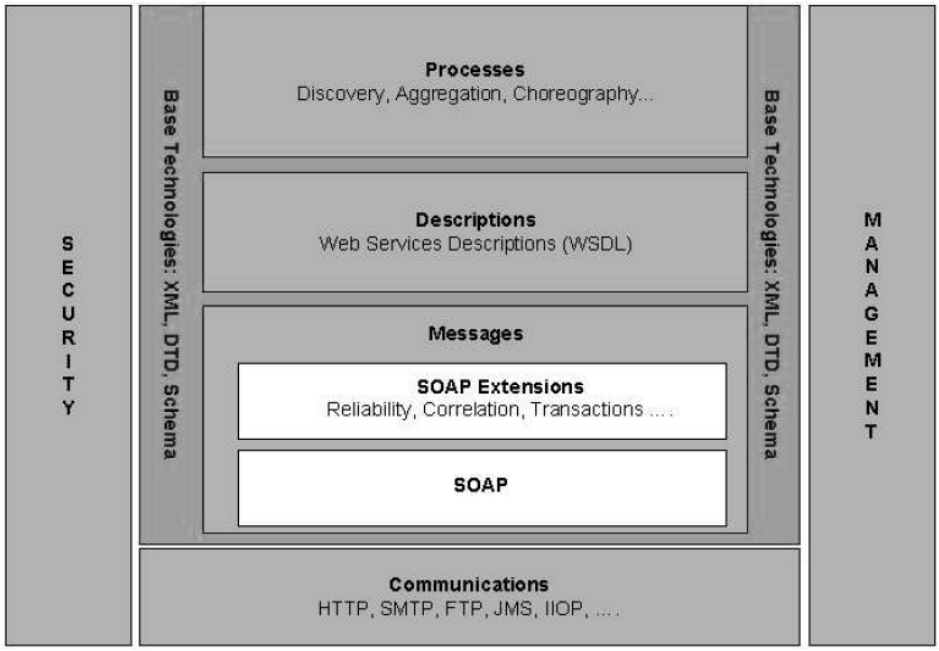

Figure 1.2: Web Services Architecture Stack [14]

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.

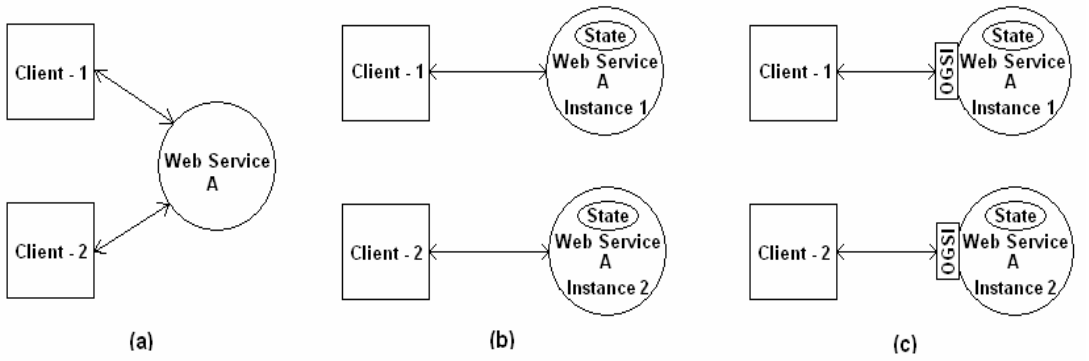

Figure 1.3: Schematic showing (a) Stateless web service (b) Stateful web service (c) Grid service.

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.

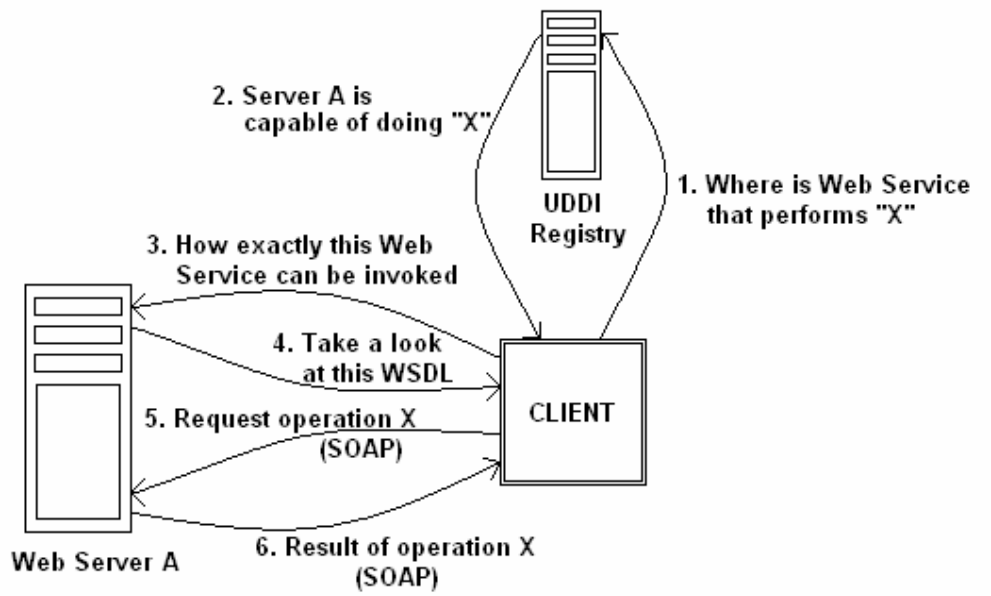

Figure 1.4: A typical web service invocation scenario

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.