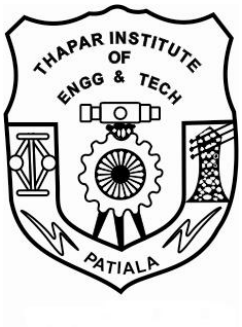
Transport Level Security in Grid Environment

A Thesis

Submitted in partial fulfillment of the
requirement for the award of degree of

Master of Engineering
in
Software Engineering



By:
Jagdish Kumar
(8043110)

Under the supervision of:
Dr. (Mrs.) Seema Bawa
Professor & Head, CSED
&
Mr. Maninder Singh
Assistant Professor, CSED

MAY 2006

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
(DEEMED UNIVERSITY)
PATIALA – 147004

ABSTRACT

Grid technology has emerged as a new way of large-scale distributed computing with high-performance orientation. Grid computing is being adopted in various areas from academic, industry research, to government use. The multi-institutional nature of grid computing poses a significant challenge in establishing a security infrastructure for the grid. In this thesis, we specifically focus on the security implementation provided by Globus toolkit 4 (GT4) and how it resolves the security issues. GT4's security implementation is based on existing web service technology, public key cryptography, and digital certificates.

Within Grid computing, it is very common to have groups of individuals and associated resources and services on different administrative domain called virtual organizations, connected together for a single purpose. Each domain has its own local security policies; thus, grid security must find a way to bridge the diverse local policies and allow inter-domain communication in a secure manner. This multi-institutional environment creates the need for a security framework that allows flexibility in communication and authentication between different domains yet remains secured from unwarranted intrusions. In order to support the above requirements, Globus toolkit incorporates a specific security component called Grid Security infrastructure or GSI which is based on standard technologies, such as TLS (formerly SSL) and secure Web Services specifications. The GSI in Globus Toolkit 4 uses message-level security, and the transport-level security.

GSI provides complete public-key system, authentication through certificates, credential delegation and single sign-on. GSI is based on public-key cryptography to guarantee privacy, integrity, and authentication. In most situations, there may not be necessary to provide all three features at once, but at the least, authentication should be performed. This thesis focuses on authentication aspect of grid security. A simple grid service is implemented and security features are added to it from authentication perspective, then experimental results are obtained for the same.

Certificate

I hereby certify that the work, which is being presented in the thesis, entitled **"Transport Level Security in Grid Environment"** in partial fulfillment of the requirements for the award of degree of Master of Engineering in Software Engineering at Computer Science and Engineering Department of Thapar Institute of Engineering and Technology (Deemed University), Patiala, is an authentic record of my own work carried out under the supervision of Dr. (Mrs.) Seema Bawa and Mr. Maninder Singh.

I have not submitted the matter presented in the thesis for the award of any other degree of this or any other university.

(Jagdish kumar)

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

(Dr. (Mrs.) Seema Bawa)
Supervisor
Professor & Head
Computer Science and Engineering Department,
Thapar Institute of Engineering & Technology, PATIALA-147004

(Maninder Singh)
Supervisor
Assistant Professor
Computer Science and Engineering Department,
Thapar Institute of Engineering & Technology, PATIALA-147004

Countersigned by

(Dr. (Mrs.) Seema Bawa)
Professor & Head
Computer Science and Engineering Department
Thapar Institute of Engineering and Technology

(Dr. T.P. Singh)
Dean of Academic Affairs
Thapar Institute of Engineering and Technology
PATIALA-147004

Acknowledgement

It is with the deepest sense of gratitude that I am reciprocating the magnanimity, which my guides Dr. Seema Bawa, Professor and Head, Computer Science and Engineering Department and Mr. Maninder Singh, Assistant Professor, Computer Science and Engineering Department has bestowed on me by providing individual guidance and support throughout the Thesis work. The successful completion of this thesis is a direct consequence of the moral and material support extended by Centre for Excellence in Grid Computing , TIET throughout this thesis.

I am also thankful to Mr. Rajesh Bhatia, P.G. Coordinator, Computer Science and Engineering Department for the motivation and inspiration that triggered me for my thesis work.

I would also like to thank all the staff members and my co-students who were always there at the need of the hour and provided with all the help and facilities, which I required for the completion of my thesis.

I am also thankful to the authors whose works I have consulted and quoted in this work. Last but not the least I would like to thank God for not letting me down at the time of crisis and showing me the silver lining in the dark clouds.

Jagdish Kumar

(8043110)

Thesis Organization

The thesis is organized as follows:

- In Chapter 1, we explain briefly about what Grid Computing is, why we need grid computing and compare it with other distributed technologies. We also give the problem definition and set objectives.
- In Chapter 2, we discuss the literature survey for security in Grids where various security issues involved in Grids are discussed in detail and the approach to deal with these issues are described .In particular we describe the authentication and authorization and explain how confidentiality, Integrity and Availability are maintained on the Grid.
- In Chapter 3, we present the state of art in grid services and describe the transition from the concept of web services to the grid services, we also discuss the WSRF in brief and its association with GT4.
- In Chapter 4, we present the procedure for establishing TIET GRID with GT4 on Linux machines and implement a simple grid service and then a secure grid service and from this the experimental results are achieved and discussed.
- In Chapter 5, we discuss grid services security considerations and conclude with current status of the Grid service security and future work.

TABLE OF CONTENTS

List of Figures

List of Tables

Chapter1
Introduction to Grid

1.1 Introduction

Grid-computing principles focus on large-scale resource sharing in distributed systems in a flexible, secure, and coordinated fashion. This dynamic coordinated sharing results in innovative applications making use of high-throughput computing for dynamic problem solving. Grid computing uses the resources of many separate computers connected by a network to solve large-scale computation problems.

The vision of the Grid computing [2, 3] is to provide high performance computing and data infrastructure supporting flexible, secure and coordinated resource sharing among dynamic collections of individuals and institutions known as "virtual organizations" (VO) [2, 3]. "Grid Computing" is rapidly emerging from the scientific and academic area to the industrial and commercial world. It is intended to offer seamless and uniform access to substantial resources without having to consider their geographical locations. Resources can be high performance supercomputers, massive storage space, sensors, satellites, software applications, and data belonging to different institutions and connected through the Internet [2, 3]. Grids can enable collaboration between several organisations [2, 3]. The Grid provides the infrastructure that enables dispersed institutions (commercial companies, universities, government institutions, and laboratories) to form virtual organisations (VOs) that share resources and collaborate for the sake of solving common problems.

1.2 Example of a Grid application

A typical example of a Grid application is "weather prediction". This involves collaboration between several partners: TV stations that produce regular weather news reports, a Satellite Company that regularly provides space images of the earth, a super computing center that rapidly analyses the images and a visualization center that produces visual interpretations of the weather analysis (Figure 1.1). The smooth running of this project for the timely production of regular weather reports crucially depends on

appropriate schemas for securely sharing, exchanging, and coordinating information between these partners.

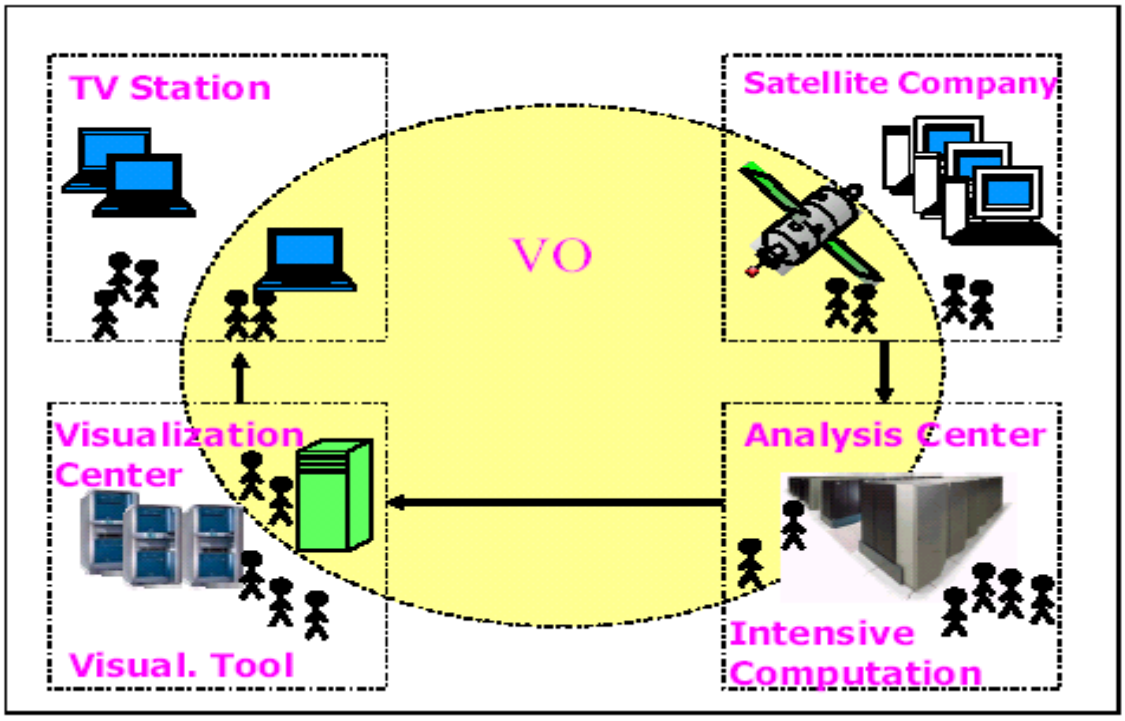


Figure 1.1 Weather prediction using Grid [6]

The power of Grid is particularly useful in arenas involved in intensive processing such as life science research [6], financial modeling [6], industrial design [6], and graphics rendering [6]. Many governments have recently initiated special programmes to support the Grid: The benefits of having partnerships between institutions to achieve ambitious projects have been well recognized.

1.3 Need of Grid Computing

In the last few years, a crucial gap has developed between the advance of networking capability (the bits per second a network can handle) and microprocessor speed (based on the number of transistors per integrated circuit). Networking capability essentially doubles every nine months today, although historically this growth was much slower. And Moore's Law dictates that the number of transistors per integrated circuit still

doubles every 18 months. Therein lies the problem. Moore's Law is slow compared with the advancement in network capability. If you accept as given that core networking technology now accelerates at a much faster rate than advances in microprocessor speeds, then it becomes apparent that in order to take advantage of the advances in networking, a more efficient way of harnessing microprocessor capacity is required. This new point of view changes the historical trade-off between networking and processing costs. Similar arguments apply to bulk storage. Grid computing is the means to address this gap, this change in the traditional trade-offs, by tying together distributed resources to form a single virtual computer [3].

In 1998, it was stated that a computational Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities [1]. This definition was primarily centered on the computational aspects of Grids. Later iterations broadened this definition with more focus on coordinated resource sharing and problem solving in multi-institutional virtual organizations. Grid computing differentiates itself from other distributed computing technologies through an increased focus on resource sharing, co-ordination, manageability, and high performance. The sharing of resources, ranging from simple file transfers to complex and collaborative problem solving, is accomplished under controlled and well-defined conditions and policies. In this context, the critical problems are resource discovery, authentication, authorization, and access mechanisms.

Grid computing offers a model for solving massive computational problems by making use of the unused resources (CPU cycles and/or disk storage) of large numbers of disparate, often desktop, computers treated as a virtual cluster embedded in a distributed telecommunications infrastructure. Grid computing's focus on the ability to support computation across administrative domains sets it apart from traditional computer clusters or traditional distributed computing. Grids offer a way to solve grand challenge problems like financial modeling, earthquake simulation, climate/weather modeling etc. Grids offer a way of using the information technology resources optimally inside an organization. They also provide a means for offering information technology as a utility bureau for clients, who pay only for what they use, as with electricity or water.

Grid computing has the design goal of solving problems too big for any single supercomputer, whilst retaining the flexibility to work on multiple smaller problems. Thus Grid computing provides a multi-user environment. It involves sharing heterogeneous resources (based on different platforms, hardware/software architectures, and computer languages), located in different places belonging to different administrative domains over a network using open standards. In short, it involves virtualizing computing resources.

1.4 Characteristics of Grid

The following are the characteristics of a Grid [1]:

1) Coordinates resources that are not under centralized control.

2) Uses standard, open, general-purpose protocols and interfaces.

3) Delivers non-trivial qualities of service.

1.5 Grid Types

Grids can be classified on the basis of two parameters

- Grid functionality
- Grid topology

1.5.1 Functional classification

Grid systems can be classified into two categories. Real Grids may be a combination of two of these types. The two categories of Grid systems are described below.

- Computational Grid
- Data Grid

1.5.1.1 Computational Grid

Computational grids can be recognized by these primary characteristics:

- Made up of clusters of clusters.
- Enables CPU scavenging to better utilize resources.

- Provides the computational power to process large-scale jobs to satisfy the business requirement for instant access to resources on demand

1.5.1.2 Data Grid

Data grids focus on providing secure access to distributed, heterogeneous pools of data. Data grids also harness data, storage, and network resources located in distinct administrative domains, respect local and global policies governing how data can be used.

1.5.2 Topology Classification

Grids can be built in all sizes, ranging from just a few machines in a department to groups of machines organized as hierarchy spanning the world. Grids can be classified into three categories according to the topology of Grid.

- IntraGrid
- ExtraGrid
- InterGrid

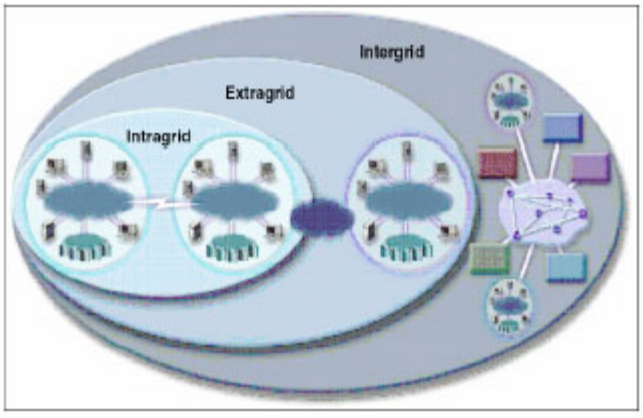


Figure. 1.2 Grid Types [8]

1.5.2.1 Intra Grid

- There is a single organization.
- Partner integration is not there.
- Only one single cluster.

1.5.2.2 ExtraGrid

- There are multiple organizations.
- Partner integration is there.
- Multiple clusters

1.5.2.3 InterGrid

- There are many organizations.
- There are multiple partners
- Many multiple clusters

1.6 Virtual Organization

A virtual organization (VO) is a dynamic group of individuals, groups, or organizations who define the conditions and rules for sharing resources. The concept of the VO is the key to Grid computing. These are some of the common characteristics that typically exist among participants of a VO:

- Concerns and requirements exist concerning resource sharing.
- Resource sharing is conditional, time-bound, and rules-driven.
- The collection of participating individuals and/or institutions is dynamic.
- Sharing relationship among participants is peer-to-peer in nature.
- Resource sharing is based on an open and well-defined set of interactions and access rules.

All VOs share some characteristics and issues, including common concerns and requirements that may vary in size, scope, duration, sociology, and structure. The members of any VO negotiate the sharing of resources based upon the rules and conditions defined by the VO, and the members then share the resources in the VO's constructed resource pool. Assigning users, resources, and organizations from different domains to a VO is one of the key technical challenges in Grid computing. This task includes identification and application of appropriate resource-sharing methods, rules and

conditions for member assignment, security delegation, and access control among the participants.

1.7 Comparison with other distributed computing technologies

Grid computing has recently enjoyed an increase in popularity as a distributed computing architecture. As Grid computing matures, the application of the technology in additional areas will increase. Grid computing can be differentiated from almost all distributed computing paradigms by this defining characteristic: The essence of Grid computing lies in the efficient and optimal utilization of a wide range of heterogeneous, loosely coupled resources in an organization tied to sophisticated workload management capabilities or information virtualization [3].

1.7.1 Comparison with Cluster computing

Grid computing is often confused with cluster computing. The key difference is that the resources which comprise the Grid are not all within the same administrative domain. Grids consist of heterogeneous resources. Cluster computing is primarily concerned with computational resources, Grid computing integrates storage, networking, & computation resources. Clusters usually contain a single type of processor and operating system; Grids can contain machines from different vendors running various operating systems. Grids are dynamic by their nature. Clusters typically contain a static number of processors and resources; resources come and go on the Grid. Resources are provisioned onto and removed from the Grid on an ongoing basis. Grids are inherently distributed over a local,metropolitan, or wide-area network. Usually, clusters are physically contained in the same complex in a single location; Grids can be (and are) located everywhere. Cluster interconnection technology delivers extremely low network latency, which can cause problems if clusters are not close together. Grids offer increased scalability. Physical proximity and network latency limit the ability of clusters to scale out; due to their dynamic nature, Grids offer the promise of high scalability.

1.7.2 Comparison with CORBA

Of all distributed computing environments, CORBA probably shares more surface level similarities with Grid computing than the others. This is due to the strategic relationship between Grid computing and Web services in the Open Grid Services Architecture (OGSA). Both are based on the concept of service-oriented architecture (SOA).

A key distinction between CORBA and Grid computing is that CORBA assumes object orientation, but Grid computing does not. In CORBA, every entity is an object and it supports mechanisms such as inheritance and polymorphism. In OGSA, there are similarities to some object concepts, but there isn't a presumption of object-oriented implementation in the architecture. The architecture is message oriented; object orientation is an implementation concept. However, the use of a formal definition language (such as WSDL, Web Services Definition Language) in WSRF (Web Services Resource Framework) means that interfaces and interactions are just as precisely defined as in CORBA, sharing one of the major software engineering benefits also exhibited by object-oriented design.

1.7.3 Comparison with P2P

The hallmark of a P2P system is that it lacks a central point of management; this makes it ideal for providing anonymity and offers some protection from being traced. Grid environments, on the other hand, usually have some form of centralized management and security (for instance, in resource management or workload scheduling). This lack of centralization in P2P environments carries two important consequences:

1) P2P systems are generally far more scalable than Grid computing systems. Even when you strike a balance between control and distribution of responsibilities, Grid computing systems are inherently not as scalable as P2P systems.

2) P2P systems are generally more tolerant of single-point failures than Grid computing systems. Although Grids are much more resilient than tightly coupled distributed systems, a Grid inevitably includes some key elements that can become single points of failure. This means that the key to building Grid computing systems is finding a balance between decentralization and manageability.

1.8 Security issues in Grid Environment

Grid applications are characterized by the coordinated use of resources from different administrative domains. Figure 1.3 indicates this situation by showing the policies and platforms in each domain. Each site in the VO is independently administered and has its own local security solutions such as Kerberos and PKI. These solutions are built on top of different platforms such as UNIX [27], Windows [25] and OS2 [24].

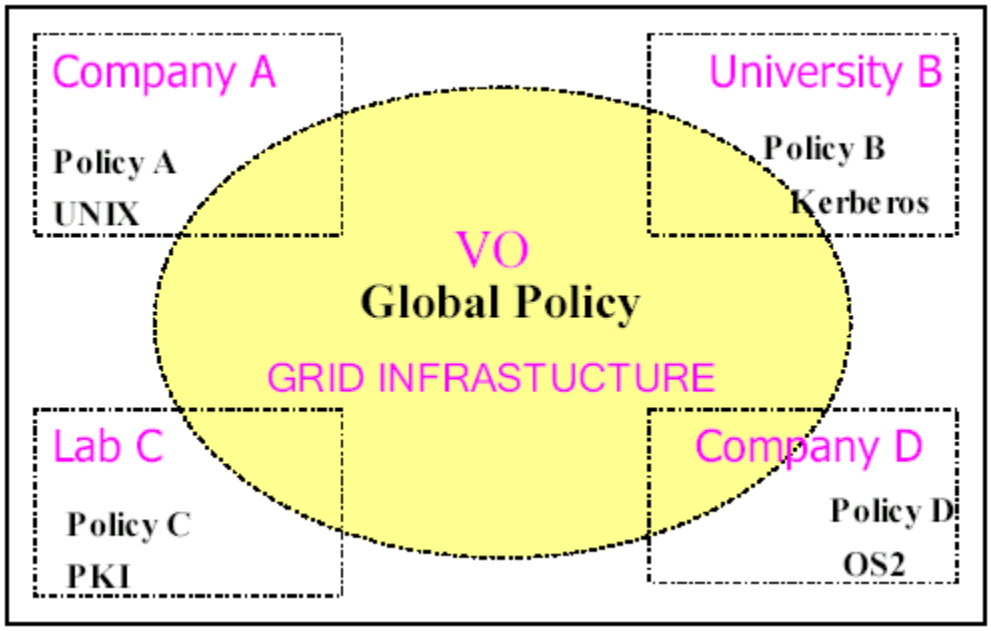


Figure 1.3 *Reconcile local policies with Global policy[11]*

When these institutions are brought together to collaborate on a common project in this heterogeneous environment, many security problems arise:

1.8.1 Interoperability

Interoperability is a key issue on the Grid. It is impractical to change the security mechanisms at each site in the Grid due to technical, financial and political reasons [11].

Thus, the security of the Grid project must be able to interoperate with the local security solutions at different levels:

1.8.1.1 Policy level

Each partner in the Grid has its own security policy (Figure 1.3), which is carefully tailored to maximize the protection of its valuable resources. The main issues to be addressed are:

- How to reconcile global security policy with local security policy.
- How conflicts between local and global policy can be solved, in other words which policy will apply, local or global.

1.8.1.2 Authentication level

Grid requires mechanisms for identifying users from one security domain to another. For example, the identity of a user from company **A** (**U.A**) and his credential as expressed in Policy **A** are meaningless in the other sites. Therefore, how does **U.A** authenticate (i.e. UNIX login) to site **B** to access resource (**R.B**) (i.e. Kerberos)?

1.8.1.3 Authorization level

Access control mechanisms used vary from one site to another depending on the type and value of the resource accommodated. For example, site **A** may use an Access Control List (ACL) [1] or a Role Based Access Control (RBAC) [1] as mechanisms in order to gain access to its resources. The first problem is how to determine whether a user, **U.A,** authenticated in site **B,** is allowed access to resource, **R.B** in **B**. The second is who decides what the access rights of **U.A** are.

1.8.2 Scalability

The number of users and resources in the Grid environment are dynamic. New users or resources can be added/removed to the project as required. For example, a project, may involves 1800 users from 150 institutions in 32 countries. Thus, a scalable way to manage users' authentication and their access rights to access project resources is required.

1.8.3 Confidentiality and Integrity issues

On the Grid, users transmit data over the Internet and access remote data resources that may be very sensitive. Moreover, Grid users can run programs on remote sites. Therefore, confidentiality and integrity are required to:

- Protect transmitted data over a public network such as the Internet
- Ensure the privacy and accuracy of the results of programs executed on remote sites.
- Ensure the secrecy and correctness of the shared data resources.

1.8.4 Trust

Scientists and commercial companies want to know whom they are trusting with their data and commodities. The question that arises: Who to trust individuals/ sites/ third parties.

1.9 Globus

The Globus [8, 22] project is a U.S. multi-institutional research effort that seeks to enable the construction of computational Grids. A computational Grid, in this context, is a hardware and software infrastructure that provides dependable, consistent, and pervasive access to high-end computational capabilities, despite that, toolkit consists of a set of components that implement basic services, such as security, resource allocation, resource management, and communications. Globus can be viewed as a Grid computing framework based on a set of APIs to the underlying services. Globus provides application developers with a pragmatic means of implementing a range of services to provide a wide-area application execution environment.

It provides a software infrastructure that enables applications to handle distributed heterogeneous computing resources as a single virtual machine. Globus provides basic services and capabilities that are required to construct a computational Grid. Globus provides a bag of services which developers of specific tools or applications can use to meet their own particular needs. This methodology is only possible when the services are

distinct and have well-defined interfaces (APIs) that can be incorporated into applications or tools in an incremental fashion. Globus is constructed as a layered architecture in which high-level global services are built upon essential low-level core local services.

1.9.1 Overview of Globus Toolkit 4

Globus Toolkit 4 is a collection of open-source components. Many of these are based on existing standards, while others are based on (and in some cases driving) evolving standards. Version 4 of the toolkit is the first version to support Web service based implementations of many of its components. (Version 3 had included an OGSI implementation of some components, and Version 2 was not service based at all.) Though many components have Web service based implementations, some do not, and for compatibility and migration reasons, some have both implementations. Globus Toolkit 4 provides components in the following categories:

Common runtime components

- Security
- Data management
- Information services
- Execution management

Web Service Components					Non Web Service Components	
Common Runtime Component	***Java WS-Core***	***C WS-Core***	***Python WS-CORE***		***C Common Liberaries***	***Extensible IO- XIO***
Security Component	***WS-Authentication & Authorization***	***Community Authorization Service***	***Delegation Service***		***Pre-WS-Authentication & Authorization***	***Credential Mgmt***
Data Mgmt	***RFT***	***OGSA DAI***	***Data Replication Service***		***GridFTP***	***Replica Location Service***
Monitoring & Discovery Services	***Index Service***	***Trigger Service***	***Aggregator Framework***	***WebMDS***	***MDS2***	
ExecutionMgmt	***WS-GRAM***	***Community Scheduler Framework***	***Grid Telecorp Control Protocol***	***Workspace Mgmt Service***	***Pre WS GRAM***	

Table 1.1 List of components in Globus Toolkit 4

1.9.2 Standards for Grid Environment

Grid [10] computing assumes and/or requires technologies that include:

- Support for executing programs on a variety of platforms
- A secure infrastructure
- Data movement/replication/federation
- Resource discovery
- Resource management

For each of these areas, there are a variety of technologies available that could be used to address them. We will look at just a few of the standards that could be considered when architecting a grid-based solution.

Standards bodies that are involved in areas related to grid computing include:

- Global Grid Forum (GGF)[21]
- Organization for the Advancement of Structured Information Standards (OASIS)
- World Wide Web Consortium (W3C)[18]
- Distributed Management Task Force (DMTF)
- Web Services Interoperability Organization (WS-I)

1.9.2.1 OGSA

The Global Grid Forum [21] has published the Open Grid Service Architecture [8]. To address the requirements of grid computing in an open and standard way, requires a framework for distributed systems that support integration, virtualization, and management. Such a framework requires a core set of interfaces, expected behaviors, resource models, and bindings. OGSA defines requirements for these core capabilities and thus provides general reference architecture for grid computing environments. It identifies the components and functions that are useful if not required for a grid environment. Though it does not go to the level of detail such as defining programmatic interfaces or other aspects that would guarantee interoperability between implementations, it can be used to identify the functions that should be included based on the requirements of the specific target environment.

1.9.2.2 OGSI

As grid computing has evolved it has become clear that a service-oriented architecture could provide many benefits in the implementation of a grid infrastructure. The Global Grid Forum extended the concepts defined in OGSA[8] to define specific interfaces to various services that would implement the functions defined by OGSA. More specifically, the Open Grid Services Interface (OGSI) defines mechanisms for creating, managing, and exchanging information among Grid services. A Grid service is a Web

service that conforms to a set of interfaces and behaviors that define how a client interacts with a Grid service. These interfaces and behaviors, along with other OGSI mechanisms associated with Grid service creation and discovery, provide the basis for a robust grid environment. OGSI provides the Web Service Definition Language (WSDL) definitions for these key interfaces. Globus Toolkit 3 included several of its core functions as Grid services conforming to OGSI.

1.9.2.3 GridFTP

GridFTP is a secure and reliable data transfer protocol providing high performance and optimized for wide-area networks that have high bandwidth. As one might guess from its name, it is based upon the Internet FTP protocol and includes extensions that make it a desirable tool in a grid environment. The GridFTP protocol specification is a proposed recommendation document in the Global Grid Forum.

GridFTP uses basic Grid security on both control (command) and data channels. Features include multiple data channels for parallel transfers, partial file transfers, third-party transfers, and more. GridFTP can be used to move files (especially large files) across a network efficiently and reliably. These files may include the executables required for an application or data to be consumed or returned by an application. Higher level services, such as data replication services, could be built on top of GridFTP.

1.9.2.4 WSRF

WSRF is being promoted and developed through work from a variety of companies, including IBM, and has been submitted to OASIS technical committees. Basically, WSRF defines a set of specifications for defining the relationship between Web services (that are normally stateless) and stateful resources. WSRF is a general term that encompasses several related proposed standards that cover:

- Resources
- Resource lifetime
- Resource properties
- Service groups (collections of resources)
- Faults

- Notifications
- Topics

As the concept of Grid services evolves, the WSRF suite of evolving standards holds great promise for the merging of Web services standards with the stateful resource management requirements of grid computing.

1.9.2.5 Web services related standards

Because Grid services are so closely related to Web services[9, 10], the plethora of standards associated with Web services also apply to Grid services. Standards commonly associate with Web services, such as:

- XML
- WSDL
- SOAP
- UDDI

In addition, there are many evolving standards related to Web Services Interoperabilty (WS-I) that also can be applied to and bring value to grid environments, standards, and proposed standards.

1.10 The Grid Problem

Grid computing has evolved into an important discipline within the computer industry by differentiating itself from distributed computing through an increased focus on resource sharing, co-ordination, manageability, and high performance. The focus on resource sharing has resulted in a set of problems associated with resource sharing among a set of individuals or groups. This sharing of resources, ranging from simple file transfers to complex and collaborative problem solving, is accomplished under controlled and well-defined conditions and policies. In this context, the critical problems are authentication, authorization, and access mechanisms, service level features, accounting, federated security, scalability, and open-ended integration.

1.11 Problem Definition

The main problem is to create a Grid service with reasonable combination of accessibility and access restrictions. The service has to be accessible to user after establishing the identity of that user that means grid service should be available only after the user is authenticated.

The goal is to create a Grid service where security is assured by common safety techniques for transport so our main focus is on developing a grid service and adding security to that service in terms of making it accessible to the user only after the user is authenticated by verifying the credentials of the user.

GT4 GSI complies with the Generic Security Service Application Programming Interface (GSS-API, RFC 2078/2743) standard. The implementation uses X.509 certificates. The basic requirements on GSI are:

- Every entity or subject accessing grid resources (user, device, resource, process or application) must have its unique identity. The identity of the subject could be represented using certificates. A trusted Certification Authority (CA) issues these certificates.
- The identity of a subject needs to be ensured or authenticated. This is provided by the TLS authentication protocol (descendant of the SSLv3 protocol). The identification information is guaranteed with the trust in CA and its signature policy.

1.12 Objective

The aim of this thesis is to investigate and implement authentication mechanism in Grid environment by implementing secure grid services [13,15] in GT4. Combining both, Web services and an intuitive way of access management is the main motivation for this thesis. The Grid is based on concepts of virtual organizations whose definitions, administrative capabilities and functionalities have rapidly evolved over the last decade. One reason for this evolution, among several others, is the drive to provide more satisfactory solutions to aspects of Grid security such as: integrity, confidentiality, availability, accountability, authorization, and authentication. The Grid solutions to these aspects depend strongly on

the definition of the trust relationship between sites and third parties, and the adoption of new advances in cryptography and PKI technology.

The main objectives are to:

- Give a brief introduction to Grid computing, the reasons for using it, and to show why in practice Grid computing is important.
- Present basic security issues in grid environment.
- Understand mechanisms for achieving fundamental security aspects such as:
    - Authentication mechanisms: user name and password, Single Sign-On [1], PKI, identity delegation [2, 4, 5].
    - Authorization mechanisms: Access Control Lists [1], Role Based Access Control [1], Identity mapping to apply local security [3, 6].
    - Confidentiality, Integrity, Availability and accountability mechanisms.
- Give an overview of the de-facto standard Globus [3] project, where Globus infrastructure introduces security access controls over resources despite their geographical distribution, and their heterogeneous nature.
- Understand the web services and the concept of state fullness and persistence in web services using WSRF
- Implement authentication mechanisms to inspect transport level security in Grid service with Globus (GT4).

Chapter 2
Literature Survey

2.1 Grid Security Requirements

As Grid resources and users are distributed and owned by different organizations, only authorized users should be allowed to access them. Security requirements [11, 12] within the Grid environment are driven by the need to support dynamic, scalable and distributed virtual organizations. The virtual organization access must be established and coordinated only through binary trust relationships that exist between the local users and their organization and the virtual organization and the user.

Grid security mechanisms address these challenges by allowing a virtual organization to be treated as a public domain overlay, which coordinates policies to allow for coordinated resource sharing. The resource is also dynamic over the virtual organization's lifetime. The Grid security mechanism should also satisfy constrains that derive from characteristics of a practical grid environment such as single sign-on (i.e. a user should authenticate once), user credentials (password, private key) protection, interoperability with local security solutions, support execution in multinational sites, uniform infrastructure, dynamic establishment of trust domains (delegation), support for secure group communication and support for multiple implementations. The main areas of security in Grid are:

- Authentication
- Authorization
- Confidentiality
- Integrity
- Availability

2.1.1 Authentication

Authentication [8] is to ascertain whether a person is bona fide. Authentication is important to authorization, confidentiality and auditing. Thus, if authentication fails, the

whole project security will fail as well. Users of a Grid project often require remote access to project valuable resources and services over the Internet. Authentication is needed because the user identity is a parameter in most access control (authorization) solutions used on the Grid [1]. Finally, authentication is crucial to accountability, because user's identity is part of security events logged in the audit trail [1].

In addition to authenticating users on the grid, resources and processes running on user's behalf may require authentication [4]. For example, users want to make sure that they are communicating with a genuine resource before sending confidential data.

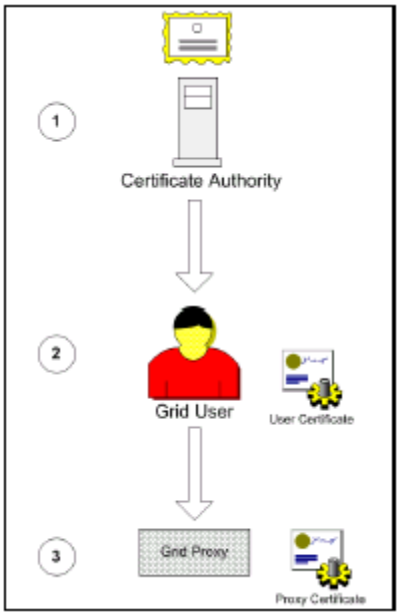


Figure 2.1 Authentication Process [8]

The hierarchy is as follows:

1. The remote grid resource trusts the CA. The remote grid resource trusts the CA because it placed the CA's certificate in /etc/grid-security/certificates.

2. The remote grid resource can authenticate the user certificate because the CA digitally signs it.

3. The remote grid resource can authenticate the user proxy certificate because the user certificate digitally signs it.

2.1.1.1 Mutual Authentication

Grid relies on trust between parties. An authorized person is trusted to use a service. You cannot ascertain authorization until the person in question is authenticated unequivocally as that person and not an impostor. It is also necessary for a person to ascertain that a service is the one they are looking for and not an impostor posing as that service, before handing over potentially important material. Therefore the need for mutual authentication provision is vitally important within any grid software.

2.1.1.2 Third Parties

Grids make use of third party Certification Authorities (CA's) to enable mutual authentication. A CA is used to validate people who apply for certificates for use in mutual authentication. They ensure that a person is bona fide before handing them a certificate. That certificate is signed by the CA to state, which CA issued it. Therefore in the mutual authentication process it is a trusted third party that is in effect vouching for who a person is. This is a somewhat simplified view of what a CA does, but it suffices to say that it is a very important requirement that the role third parties play, be considered carefully in the design of any grid.

Mutual authentication process can be explained using following steps:

- A and B are two parties: Both need to trust each others' CA.
- A → B (A establishes connection to B and gives his certificate (name,pub.Key) to B).
- B makes sure that it can trust CA of A.
- B generates random message→ A and asks it to encrypt it.
- A encrypts it and send to B
- B decrypts using A's public key. If the msg. is same as what B has sent, then A is who it is claiming to be.

2.1.1.3 Use of Proxies

A proxy is a temporary credential used to make use of services without the need for a user to individually authorize each request. They are an important component of the mutual authentication process; though they are recognized to have security breach

implications. They are however found to enhance the overall usability of systems significantly enough to make it necessary to provide a security provision for them.

A typical Grid project consists of a dynamic number of resources. A project user can initiate a job request that involves the coordinated use of these different resources at different sites. However, this raises two problems:

- **Convenience:** it is impractical for the user to authenticate multiple times to access different resources [2].
- **Vulnerability:** the user will have to sign a challenge for each authentication with his private key. This provides an opportunity for an attacker to collect challenges and their corresponding cipher texts in order to recover the private key [5].

These problems are solved using proxy credentials [2, 5, 11]. A proxy certificate is a special X.509 v3 [16] certificate that is signed by the private key of a Grid entity [4, 5]. It allows processes running on user's behalf to authenticate the user directly to Grid resources. In addition, this certificate can be used for delegation [1, 5] that is the process of passing authority from one entity to another. The proxy credentials grants the bearer all/subset of the Grid entity's access right [5]. These credentials have a short lifetime usually in term of hours.

The main advantage of proxy credentials is significant speed [4, 5, 12]. Users can run jobs that require coordinated access to resources without the user's intervention. This raises new security concerns such as:

• How the proxy's key pairs are generated?
• How the proxy's private key is protected on the user's machine?
• When delegating proxy credentials to a remote Grid entity, who has control over the proxy's private key?

The first concern can be viewed from a cryptographic point of view, as the length and quality of the public key pairs are crucial. The random number generator used to generate primes for the public key should be reliable. Factorization attacks have significantly improved in the last decade [17]. When the user generates a public key 512-bit long used with RSA [26], this means it is vulnerable to factorization [17]. As a result, the attacker will be able to recover the private key from the public key [17] and sign requests on

user's behalf. Presently, there is no mechanism to detect proxy's private key compromise [5].

Another problem with proxy credentials is clock synchronization. A job may determine that it requires to access resources on a different site located in a different country. Since proxy's lifetime is short, 10 hours in Globus [12], the remote site may be 12 hours ahead. As a result, it will consider the proxy invalid as it has expired.

2.1.1.4 Design issues in Grid Authentication protocol

There are several issues that need to be taken into account when designing an authentication protocol for the Grid:

• The parties in the Grid project do not trust each other. Thus, the protocol must provide mutual authentication.

• Interoperability is fundamental. Institutions have different platforms and security solutions (i.e. Kerberos and PKI). The authentication protocol must be uniform, not platform dependent.

• Usability. For users, usability is extremely important. Access to the VO has to be seamless & transparent as accessing the local organization's resources.

• Dynamic administrating. The number of users is dynamic. New users are added and removed as required. Issuing and revoking credentials for project users should be flexible.

• Mobility. The user should not be tied to one machine. Scientists and researchers move between institutions to collaborate with each other. The authentication mechanism should allow users to be mobile.

• Organizations have to make some changes to their security policy. The authentication protocol should make these changes minimal.

• Scalability. The authentication protocol should be able to scale to the size of the virtual organisation. Currently only PKI seems to scale.

• Single sign-on [1, 2]. The project may involve many resources. It is impractical for a user to authenticate to every resource during a session. The user must be able to authenticate once per work session.

2.1.1.5 Other Alternatives

Kerberos and Secure Shell are commonly used authentication protocols. This section will show why they are not suitable for Grid environment. Here is a brief explanation of these protocols.

Kerberos: Kerberos is a TTP-aided authentication protocol based on symmetric key cryptography [1]. The Grid user and a designated trustworthy server, Key Distributed Center (KDC) [17], share a long-term secret key. The user authenticates to the resource by getting a ticket from the KDC. Kerberos splits the role of the KDC between 2 entities, the Authentication Server (AS) [1] and the Ticket Granting Server (TGS) [1], in order to limit bottlenecks and the exposure of the long-term keys. Kerberos achieves inter-organizational, authentication by sharing key servers (AS or KDC) with other organizations [12]. Kerberos meets many of the requirements such as usability and single sign-on, but when used for multi-domain authentication, several issues arise:

• Acceptability: according to [12] "using Kerberos for intersite authentication also means using it for intrasite authentication". This is not acceptable for commercial companies, because sharing key servers to allow this type of authentication means giving up control over local policy. (PKI doesn't require major changes in the local security policy of a VO site)

• Scalability: Scalability is an issue, because it is very hard to extend the protocol to multiple administrative domains due to equipments and staffing cost [12]. Also, the performance bottleneck of having a TTP seems to always limit the scalability of a system.

• Clock synchronization [1]: all clocks across VO sites will need to be synchronized. Thus the AS, TGS, and all clients and all servers must somehow have nearly the same time, or allow for the difference within the key periods and decrease security. On the Grid this is very difficult to achieve because VO resources may exist in different countries with different time zones.

• Key Revocation [1]: Kerberos has no mechanism for key revocation, but relies on the timestamps to expire. Using short timestamps give more security, but require more tickets to be issued.

• Availability: Kerberos relies on the Authentication Server and TGS server to be online. The availability becomes more important when using tickets with short lifetimes [1]. (PKI allow offline authentication)

• Kerberos does not address the delegation of access rights (tickets) [1]. (In PKI, proxy certificates can be used for delegation)

Secure Shell: Secure Shell, SSH, is another alternative used for providing remote login. SSH is based on public-key authentication and offers a secure channel over assumed reliable transport, typically TCP. It supports mutual authentication and provides confidentiality of transmitted user credentials, and can be easily deployed. SSH, however, is not suitable for Grid authentication because of:

• Usability: users have to copy the public keys for any VO site they want to access. This is not practical because not all users are security experts [1].

• Limited functionalities: SSH supports limited capabilities such as remote shell and file transfer, but not others that require authentication, such as collaborative environments and web browsers. That is why PKI currently is the most used solution in Grid project such as Globus and Unicore.

2.1.2 Grid Authorization

Resources involved in a Grid project can be extremely valuable such as supercomputers or extremely sensitive such as classified medical records. Thus controlling access to these resources is crucial to maintain their confidentiality [1], integrity [1] and availability [1]. On the Grid, authorization aims at controlling and restricting access to resources [1]. Users of a Grid project often require access to valuable and sensitive resources that do not belong to their institutions. Authorization is needed to allow legitimate Grid users to access confidential Grid information and resources. Furthermore, authorization is vital to resources' integrity because only authorized Grid users can modify data resources and equipments resources' configurations.

2.1.3 Confidentiality on the Grid

Confidentiality on the Grid is crucial. The existence of some institutions in the VO, in particular commercial companies, depends on keeping their information secret. Confidential information can include: proprietary databases, sophisticated software and intellectual property such as software source code and drug formulas and compounds. Confidentiality deals with preventing disclosure of information to unauthorized entities [1]. On the Grid, Confidentiality is needed for:

• *Ensuring the network security.* Users will remotely communicate with resources, submit jobs and upload proprietary source code to remote supercomputers over the Internet. This means traffic between users and resources is a point of attack.
• *Ensuring the privacy of data resources*. The project may involve extremely sensitive or valuable data such as patients' medical records and drug experiments databases. These data is shared on the basis that only authorized project users will have access to it.
• *Maintaining the secrecy of data processed on remote resources*. The Grid may allow its users to run programs on remote sites. The result of the program and, possibly, the source code running on the remote machine should remain confidential.

2.1.3.1 Brief overview of Encryption

Encryption is a mean of transforming plaintext into cipher text under the control of a secret key. This process is called encryption; this process can be represented as

$$\mathbf{C=E_K\,(M)}$$

Where **M** is a plaintext, **E** is the cipher algorithm, **K** is the key and **C** is the cipher text. The reverse process is called decryption or decipherment.

$$\mathbf{M = E_K(C)}$$

The current algorithms used are either symmetric [17] or asymmetric (public key cryptography) [17].

2.1.3.2 Secret key cryptography

With Secret key cryptography (symmetric encryption) both parties use the same key value to encrypt clear text into cipher and to decrypt a cipher text back into clear text. The assumption is that the key **K** must remain secret. What determines the effectiveness of the cipher

algorithm **E** is the size of the secret key length, **K**. The larger the key size, the more difficult is to break the cipher text. The most widely used symmetric encryption algorithm is Data Encryption Standard (DES) [17]. It has 56-bit key length and 64-bit block. Due to advances in computing power and its short key length, DES is currently considered not secure for high value transactions [17] and for long-term data encryption. The new successor for DES is Advanced Encryption Standard (AES) [17]. It offers 128-bit key length and 128-bit block size. A major problem with symmetric key is how to exchange the secret key between parties who do not trust/know each other such as on the Internet and in our case the Grid. Also the number of keys to be managed is huge. For instance, for a group of n users, the number of keys to be managed is n*(n-1)/2. If n = 10 users, this means number of keys to be managed is 10*9/2 = 45 keys.

2.1.3.3 Public key cryptography (PKC)

With Public key cryptography [17] (asymmetric encryption), each party has a pair of related keys: one for encryption and one for decryption. The same key cannot be used for both. The main assumption in public key cryptography is that one of the keys must remain secret (private key) and the other is made public (public key). If a message sender uses its private key to encrypt the message then any recipient who can obtain the public key can decrypt it. In contrast, if a message sender uses a public key to encrypt a message then only the owner of the private can decrypt the message. Signing is the process of encrypting with the private key. Verification is the process of decrypting with the public key. RSA [17] and EL-GAMAL [17] are two well known public key algorithms used.

The main assumption in public key cryptography is that the "Private Key must remain secret". Thus, it needs to be adequately protected. One way to protect the private key file is to encrypt it with a password. So, if it is stolen from a machine or a storage device, the user will have enough time to revoke his corresponding public key.

2.1.3.4 X.509 Certificate

X.509 version 3 is the most widely used data format for public key certificates today. It provides a uniform way for expressing identity of entities on the Grid. The Internet Engineering Task Force (IETF) [16] also standardizes it. For illustration, we give X.509 v3 certificate format in **Figure 2.2**. There are several types of certificates [17]:

• Identity certificate: contains the public key of a user and his identity together with some other information, encrypted with the secret key of the CA. This makes the certificate tamper resistant.

• Attribute certificate: contains a set of attributes of user such as his occupation, role in a company together with some other information, digitally signed under the private key of the CA. The extensions in the certificate are essential for the Grid as they allow standardising policy with respect to the use of certificates. Globus uses X.509 as the main credential for authenticating users of the Grid.

VERSION	V3
SERIAL NUMBER	77991
SIGNATURE ALGO ID	RSA & SH1
SUBJECT NAME	CN= JAGDISH, OU=ME.STUDENT,O=TIET
ISSUER NAME	CA= TRUST ME, OU-PKI, O=TRUST ME
SUBJECT PUBLIC KEY INFO	79797979454324263648…….
VALIDITY PERIOD	Not before 0/01/06, Not after 15/06/06
ISSUER PUBLIC KEY INFO	53282758327896696545…….
EXTENSIONS	KEY USAGE, POLICY RESTRICTIONS
CA SIGNATURE	859686023650868346#868469

Figure 2.2 A X.509 certificate

2.1.3.5 Certificate Authority (CA)

A CA is an institution trusted by others to guarantee for the authenticity of a public key [12, 14]. The main role of the CA is to issue digital certificates that cryptographically bind a public key to the user's identity information [17]. Signing the information using the CA's private key does this. The relying parties require the CA's public key so that they can verify the digital signature on the certificates issued by the CA [17].

The CA has many other responsibilities in addition to issuing certificates. These responsibilities include generating key pairs, revoking certificates and maintaining a Certificate Revocation List (CRL) and other revocation forms [16]. A CA can be any partner in the Grid project such as a University, Government, or a third party operating for profit

such as Verisign [28] and Entrust [19]. If the Grid project is large enough, then it might establish its own CA. A CA must be trusted only to issue valid certificate [1].

2.1.4 Integrity on the Grid

Integrity deals with the prevention of unauthorized modification of data resources and resource configurations. Integrity can be achieved at different levels:

Administration level: Hackers manage to get unauthorized access to data by exploiting known weaknesses in the systems that run the Grid project resources. This includes vulnerabilities in Operating systems, Database servers, Firewalls, Intrusion Detection Systems IDSs and Routers. Vendors respond by issuing patches and upgrades. It is therefore vital to apply all patches recommended by the vendor.

Access control level: For example in satellite institution, only authorized users are allowed to change the direction of the satellite such as administrator.

Network and configuration level: Hash and Message Authentication Code (MAC) algorithms [17] are also used to maintain integrity of data. A hash algorithm is oneway function that takes an input a message of arbitrary length, and returns an output of fixed length. A MAC takes a message and key as inputs and returns a checksum of fixed length. SHA-1 [17] and MD5 [17] are hash algorithms that produce 160-bit and 128-bit hash value respectively [17]. HMAC [17] is a MAC algorithm.

Hashing can be used to ensure the integrity of data resources by hashing backup files, archives or file systems. Once the hash value is produced, it can be encrypted and saved. In case there is a suspicion of unauthorized modification of any of these files, the hash value resulting from applying the hash function again will not be the same. Thus the modification can be detected.

Digital Signature: is a fundamental mechanism for ensuring data integrity. A digital signature is usually created by calculating the hash value of data (document, network packets) and encrypting the hash value with the private key of an entity (for performance reason). For example, the digital signature on the certificate gives assurance that the

certificate has not been modified since creation by the CA. Modifications to the certificate content can be detected because the Hash value of the certificate will be different from the value signed by the CA. Digital signature can also be applied to data resources, so Grid users can ensure that the data they are using is from the intended VO site. In communication over the public network, digital signature provides origin authentication that gives Grid users assurance of the identity of the second party they are communicating with.

Concurrency level: Since the Grid is about coordinated resource sharing, a concurrency problem will occur. A job running on the Grid may involve a file or a database update. Sub-jobs cannot simultaneously read the portion of the file concurrently being updated by another job. Therefore, an application might read partially updated data and perhaps receiving a combination of old and new data. Locking and synchronizing primitives are required to maintain data integrity [7]. Typically, these primitives are built into the files system or database to automatically prevent this.

2.1.5 Grid Availability

Most VO organisations are totally dependent on their computerized information systems and the data that they store, process and transmit. Consequently, system failure or information loss can have grave consequences on the smooth running of the Grid project. VO Organisations are also being faced with an increasingly wide and sophisticated range of threats including viruses, hacking, and denial of service attacks because resources on the Grid are connected to each other via the Internet.

Availability aims at ensuring that authorised Grid users have timely and uninterrupted access to resources on the Grid [1]. Availability could be viewed from many angles: threats to the Grid infrastructure layer, threats to the underlying operating system and supporting applications and finally threat to the network infrastructure. A major threat for the Grid is "Denial-of- Service" (DoS) [14]. Since the server is accessible through the Internet, it will be vulnerable to "Denial-of-Service" (DoS) and "Distributed-Denial-of-Service" attacks [14]. These types of attacks have been widely experienced on the Internet. For example, a DoS attack in early 2000 seriously interrupted the services of some well known Internet sites such as Amazon, eBay and Yahoo [14]. The Grid infrastructure itself has some critical

components such as the CA, the CAS server and the Grid map file. If any of these components is compromised the Grid project will fail.

CAS server: When the Grid project relies on the CAS server for granting users access to resources, the availability of the server becomes crucial. Since the server is accessible via the Internet, it will be vulnerable to a wide range of threats with different impacts on availability:

• If the CAS server is hacked, and all its content is deleted, VO users will not be able to access any resource in the VO.

• If the CAS server is infected by a worm such as Code red, the performance of the server will degrade dramatically because worms consume bandwidths in order to replicate. This will delay the response time to VO users' requests.

• CAS server is also vulnerable to network attacks such as SYN flooding that exploits the handshake operation when establishing a TCP connection. (Communication with the CAS server is done via SSL thus TCP)

• *Grid-map file*: Each VO site has a Grid map file. One way of causing interruption of services on the Grid would be by deleting the Grid-map file or the database used to map global credential to local accounts on the resource. The impact of compromising a Grid map file will affect one VO site instead of the whole project.

In Grid projects that involve data resources, replication of this data to different machines at different sites can increase data availability. Thus to stop the service the attacker need to disable all servers that host the data. Also, the DoS would consist of flooding the network with huge amount of fake requests.

2.2 Basic Security Mechanisms in GT4

The Globus Toolkit's Authentication and Authorization components provide the *de facto* standard for the "core" security software in Grid systems and applications. These software development kits (SDKs) provide programming libraries, Java classes, and essential tools for a PKI, certificate-based authentication system with single sign-on and delegation features, in either Web Services or non-Web Services frameworks. ("Delegation" means that once someone accesses a remote system, he can give the remote system permission to use his credentials to access others systems on his behalf.)

- **Pre-Web Services Authentication and Authorization** - A non-Web services implementation of the Grid Security Infrastructure (GSI), containing the core libraries and tools needed to secure applications using GSI mechanisms
- **Web Services Authentication and Authorization** - A Web services implementation of the Grid Security Infrastructure (GSI), containing the core libraries and tools needed to secure applications using GSI mechanisms.

2.2.1 Transport-level and Message-level Security

GT4.0 supports both message-level and transport-level security "message-level security" means, support for the WS-Security standard and the WSSecureConversation specification to provide message protection for SOAP messages. "transport-level security" we mean authentication via TLS[5] with support for X.509proxy certificates.
GT4.0's support for message-level security is important as it allows us to comply with the WS-Interoperability Basic Security Profile.

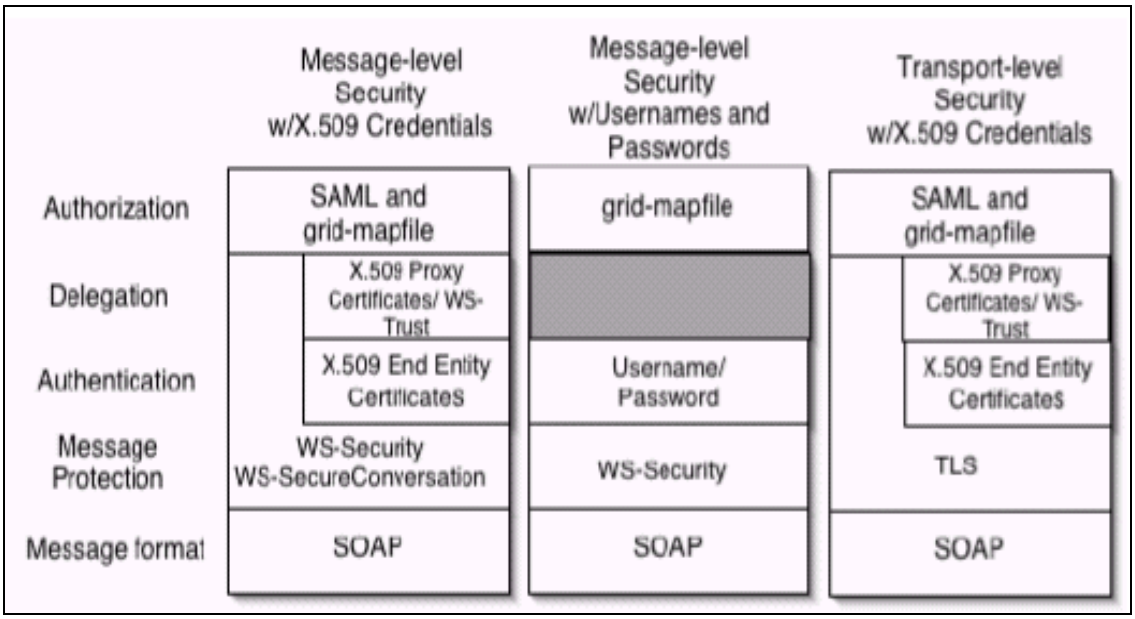


Figure 2.3 An Overview of GT4 – GSI [22]

As shown in Figure 2.3 , GSI may be thought of as being composed of four distinct functions: message protection, authentication, delegation, and authorization. Implementations of different standards are used to provide each of these functions:

- TLS (transport-level), WS-Security and WS-SecureConversation (message level) are used as message protection mechanisms in combination with SOAP.
- . X.509 End Entity Certificates or Username and Password are used as authentication credentials.
- X.509 Proxy Certificates and WS-Trust are used for delegation
- SAML assertions are used for authorization

GSI allows us to enable security at two levels: the *transport* level or the *message* level. To explain the difference between these two levels, let's suppose we want our communication to be private. If we use transport-level security, as shown **in Figure 2.4,** "Transport-level security", then the complete communication (all the information exchanged between the client and the server) would be encrypted. If we use message-level security, as shown in **Figure 2.5,** "Message-level security", then only the *content* of the SOAP message is encrypted, while the rest of the SOAP message is left unencrypted.

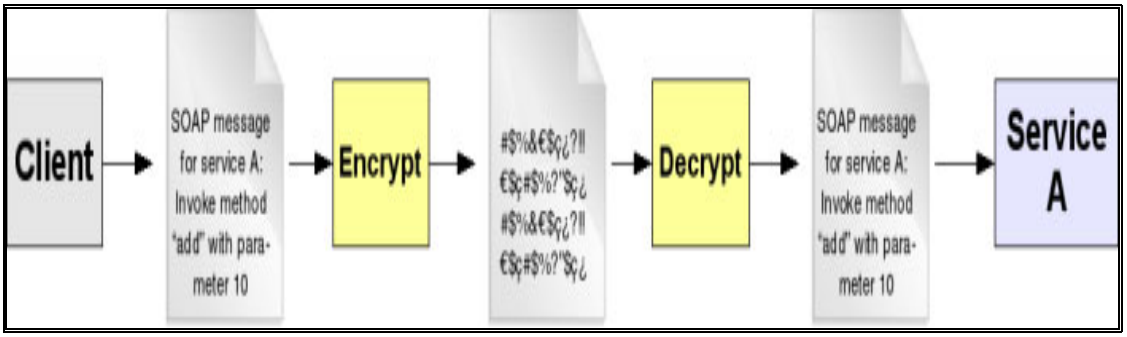


Figure 2.4 Transport-level security[20]

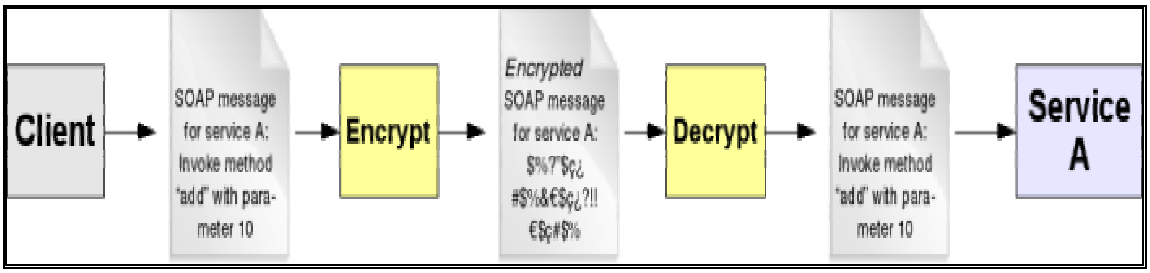


Figure 2.5 Message-level security[20]

Both transport-level and message-level security in GSI are based on public-key cryptography and, therefore, can guarantee privacy, integrity, and authentication. However, not all communications need to have those three features all at once. In general, a GSI secure conversation must *at least* be authenticated. Integrity is usually desirable, but can be disabled. Encryption can also be activated to ensure privacy.

2.2.2 Message-level vs. Transport-level performance

Transport-level security has been around for a long time and, in fact, chances are that you've already used it when browsing the Web, since secure websites rely on transport-level security. Message-level security in Web Services is relatively new and, although it offers more features than transport-level security (e.g. a better integration with Web Services standards), its performance still leaves a bit to be desired. So, even though we would ideally like to use message-level security for everything (because of its feature-rich goodness), we will sometimes have to consider using transport-level security if performance is an issue.

GSI offers two message-level protection schemes, and one transport-level scheme. The differences between these three schemes are highlighted in Table 2.1, "Comparison of transport-level and message-level security".

- **GSI Secure Message**: Provides message-level security and is based on the WS-Security standard.
- **GSI Secure Conversation**: Provides message-level security and is based on the WS-SecureConversation specification. When this method is chosen, a *secure context* is first established between the client and the server. After an initial exchange of messages to establish the context, all the following messages can

reuse that context, resulting in a better performance than GSI Secure Message (if the overhead of setting up the context is acceptable). Furthermore, GSI Secure Conversation is the only scheme that supports credential delegation (explained further on).

- **GSI Transport**: Provides transport-level security by using TLS (formerly known as SSL).

These schemes are *not* mutually exclusive. For example, we might choose to use GSI Secure Conversation because our application requires delegation, and then add GSI Transport on top of that because we want to encrypt the complete communication (not just a part of the SOAP message). Note that this doesn't result in any redundancy, since we could configure GSI Transport to use encryption and GSI Secure Conversation to *not* use encryption.

	GSI Secure Conversation	**GSI Secure Message**	**GSI Transport**
Technology	WS-SecureConversation	WS-Security	TLS
Privacy (Encrypted)	YES	YES	YES
Integrity (Signed)	YES	YES	YES
Anonymous authentication	YES	NO	YES
Delegation	YES	NO	NO
Performance	Good if sending many messages	Good if sending few messages	Best

Table 2.1 Comparison of transport-level and message-level security

Chapter 3
State of Art in Grid Services

3.1 State of the art review

Web services [8, 18] are a platform- independent way to establish communication between two applications connected through a network. Here the maintenance of policies is even more expensive. A distributed application can use many different functions and every single one needs to be secured in a proper way. It would be a hard task for a human to adjust all security levels properly because new functions need to be added while others are obsolete and not used any more. Thus it is quite possible to implement a company's applications as Web services and make them accessible through published descriptions. The advantages of such architecture are obvious: Services are completely independent of implementation or operating system and useable from everywhere within the entire network. Developers are encouraged to use the provided functionality without further knowledge of the involved code. Each Web service is registered at a centralized spot which makes service discovery much easier. Another trend in the development of computer systems is to adapt the human way of thinking, to address security aspects like access rights or judgment of the other parties intent.

This section introduces Web service technologies and explains how to invoke, publish and provide Web services in a distributed environment.

3.1.1 The Notion of the Web Service

Web Services - a standardized way of integrating Web-based applications, are the fundamental building blocks towards the distributed computing paradigm on the Internet, provide a distributed computing capability for integrating extremely heterogeneous applications over the networked environment. Web Services are operated using non-proprietary standards and specifications that are completely independent of programming language, operating system and hardware that facilitates the delivery of business applications as a service accessible to anyone, anytime, anywhere using any platform [18]. Thus, Web Services enable organizations to achieve the goals of distributed

computing [18] by promoting the loosely coupling between the service consumer and service provider.

The purpose of a Web Service is to provide some functionality on behalf of service provider entity to implement a particular service and service requester entity to make use of a provider entity's Web Service [18] where the requester entity uses a requester agent to exchange messages with the provider entity's provider agent.

3.1.2 Example of a Web Service

For example, let's suppose I keep a database with up-to-date information about weather in the United States, and I want to distribute that information to anyone in the world. To do so, I could *publish* the weather information through a Web Service that, given a ZIP code, will provide the weather information for that ZIP code.

The *clients* (programs that want to access the weather information) would then contact the *Web Service* (in the *server*), and send a *service request* asking for the weather information. The server would return the forecast through a *service response*.

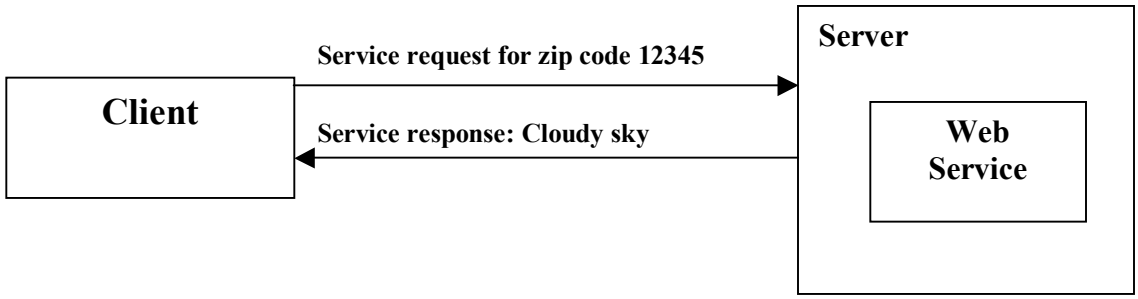


Figure 3.1 Web Service Example

3.1.3 Web Service Properties

A typical service has the following six properties [18]. Functional and non- functional: Services are described with a description language that provides functional and non-functional properties. The functional properties represent the operational level characteristics that define the overall behavior of the service. The non-functional properties consists of the quality attributes of services such as authentication,

authorization, auditing, performance, cost, accuracy, integrity, reliability, scalability, availability, interactivity support, etc.

3.1.3.1 State

Services can be divided into stateless or stateful according to the state properties. Stateless services can be invoked repeatedly without having to maintain context or state, i.e., an instance of service is stateless if it cannot retain prior events.

In case of stateful services, it can retain its prior actions, i.e., a stateful service maintains some state between different operation invocations issued by the same or different clients or applications. For example, in business process information services, there is a need of stateful interactions involving exchange of messages between partners, where the state of business process retains to undertake series of interrelated tasks to finish business process: purchase order, bank transfer, taxation, acknowledgement, and shipping notifications tasks, etc.

3.1.3.2 Transient-ness

Services can be transient or non-transient. A transient service instance is one that can be created and destroyed, usually created for specific clients and do not outlive its clients. A non-transient or persistent service instance - without the concept of service creation and destruction, outlives its clients. In this regard, normally Web Services are usually considered of as non-transient and stateless services where as Grid Services is generally thought as transient and stateful services.

3.1.3.3 Complexity

Services can vary in function from simple requests to complex systems where the system access and combines information from multiple sources. Simple service may have complicated realizations of the interactions among the trading partners.

3.1.3.4 Granularity

The concept of granularity can be applied in two ways: coarse grained and fine-grained Services [18]. The services itself can be coarse grained or fine grained that refers to how

much functionality the service covers. For example, if developers need to create a banking application for manipulating both checking account and saving account should be taken into consideration. In this case, the developer has two choices while creating a service with above functions namely. They could create coarse grained service having both the functionality in one interface or they could create two fine-grained services to manipulate checking and saving accounts respectively.

3.1.3.5 Synchronicity

Services can be distinguished between two programming styles for services: synchronous or Remote Procedure Call (RPC)-style versus asynchronous or message (document)-style. Synchronous services are method driven services where client program express their services as a method call with appropriate set of arguments and server returns a response message containing the results of the program executed. RPC-style service requires tightly coupled model of communication between the client and server to maintain the bilateral communication between clients and server. The Common Object Request Broker Architecture (CORBA) and Distributed Component Object Model (DCOM) are examples having RPC style capabilities. On the other hand, asynchronous services document style or message driven services where client sends an entire document such as purchase order, rather than discrete set of parameters. The service accepts the entire document that it processes and may or may not return a result message. Asynchronous service promotes a loosely coupling between the clients and server and is useful where the client does not require (or expect) an immediate response and process-oriented services.

3.1.4 Web services Invocation

Web service is a platform and implementation independent software component; can be invoked through a declared Application Programming Interface (API); described, published and discovered through a standard mechanism over the network; may be composed with other services. They are self-contained, self-describing, modular applications that can be published, located, and invoked across. A Web Services is a software system designed to support interoperable machine-to-machine interaction over a

network. It has an interface described in a machine-processable format (specifically Web Services Description Language (WSDL) [18]). Moreover, Universal Description Discovery and Integration (UDDI) provides a mechanism for clients to dynamically find other Web Services across the network. Therefore XML, SOAP, WSDL, and UDDI are known as core Web Services technology. For example, the following Figure 3.2 shows the typical scenario of Web Services. First, client queries registry to locate service. Second, registry refers to WSDL document. Third, client accesses WSDL. Fourth, WSDL provides data for interaction with Web Service. Fifth, client sends request toWeb Services via SOAP. Finally, Web Services replies via SOAP

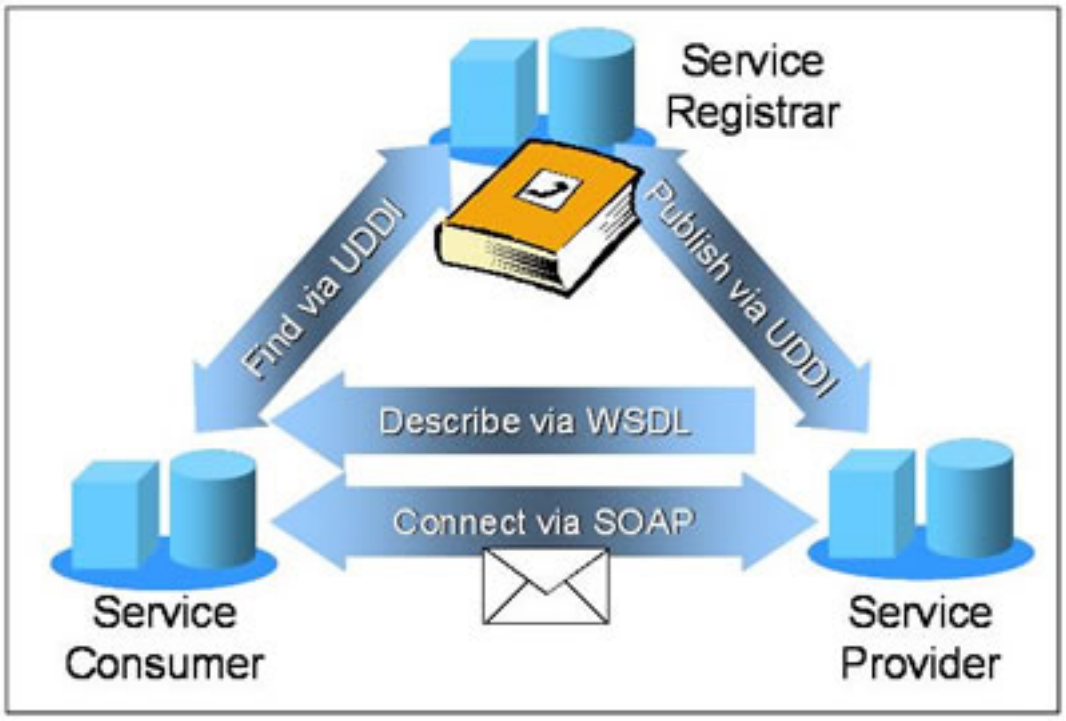


Figure 3.2 Web Services Invocation [20]

3.1.5 Advantages & Disadvantages of web services

Web Services have certain advantages over other technologies:

- Web Services are platform-independent and language-independent, since they use standard XML languages. This means that my client program can be programmed

in C++ and running under Windows, while the Web Service is programmed in Java and running under Linux.

- Most Web Services use HTTP for transmitting messages (such as the service request and response). This is a major advantage if you want to build an Internet-scale application, since most of the Internet's proxies and firewalls won't mess with HTTP traffic (unlike CORBA, which usually has trouble with firewalls).

Of course, Web Services also have some disadvantages:

- Overhead. Transmitting all your data in XML is obviously not as efficient as using a proprietary binary code. What you win in portability, you lose in efficiency. Even so, this overhead is usually acceptable for most applications, but you will probably never find a critical real-time application that uses Web Services.
- Lack of versatility. Currently, Web Services are not very versatile, since they only allow for some very basic forms of service invocation. CORBA, for example, offers programmers a lot of supporting services (such as persistency, notifications, lifecycle management, transactions, etc.). Fortunately, there are a lot of emerging Web services specifications (including WSRF) that are helping to make Web services more and more versatile.

3.2 What is a Grid service

A service interface associated with a grid resource is known as a Grid service. A resource and its state is controlled and managed via Grid services in a Grid environment. A Grid service may require access to more than one resource or vice versa. It is also possible that multiple Grid services access the same resource or a Grid service can create a new instance of a resource every time it is invoked.

Various grid resources may require interacting and integrating with each other depending on business requirements. It is most likely that the resources are hosted in a technologically heterogeneous environment. Therefore, a framework is required that abstracts environment-specific resource implementation details .A service oriented architecture (SOA) provides such a framework. It follows that an open standards

compliant SOA architecture would make it easier to integrate heterogeneous resources and various layers of the grid architecture. Such architecture would help us achieve distributed resource sharing across heterogeneous and dynamic virtual organizations, that is, grid computing.

The Global Grid Forum (GGF) has adopted an SOA principles based Open Grid Services Architecture (OGSA) that provides a framework for implementing a Grid.

All of the resources (physical or logical) in an OGSA-compliant grid are modeled as Grid services. These Grid services are built on top of a SOA leveraging WEB services technology. This enables a Grid service to use the capabilities of the Web services messaging model, service descriptions, and discovery. Various Web services standards have evolved to enable secure and reliable Web services transactions. The choice of Web-services technology to implement the OGSA-compliant Grid services leverages investment in Web-services architecture and its standards.

3.2.1 Grid services vs. Web services

Although Grid services are implemented using Web-services technology, there is a fundamental difference between a Grid service and a Web-service. A Web-service addresses the issue of discovery and invocation of persistent services. A Web Services Description Language (WSDL) compliant document points to a location that hosts the Web service. A Grid service addresses the issue of a virtual resource and its state management. A grid is a dynamic environment. Hence, a Grid service can be transient rather than persistent. A Grid service can be dynamically created and destroyed, unlike a Web service, which is often presumed available if its corresponding WSDL file is accessible to its client. Web services also typically out live all their clients.

This has significant implications for how Grid services are managed, named, discovered, and used. The OGSA model adopts a Factory design pattern to create transient Grid services. Thus, an OGSA Grid service is a potentially transient Web service based on grid protocols using WSDL.

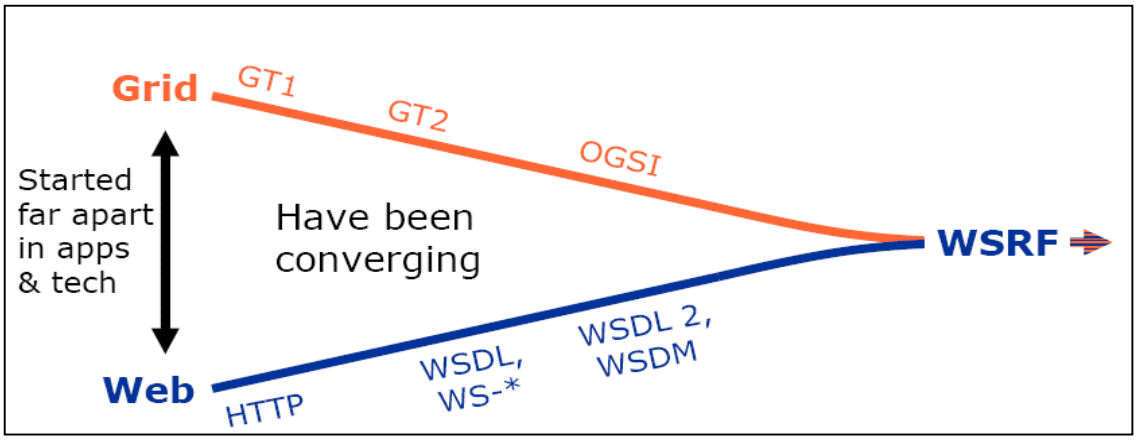


Figure 3.3 Grid Services v/s Web Services [20]

Writing and deploying a WSRF Web Service [13] needs to follow five simple steps.

1. **Define the service's interface**. This is done with *WSDL*
2. **Implement the service**. This is done with *Java*.
3. **Define the deployment parameters**. This is done with *WSDD* and *JNDI*
4. **Compile everything and generate a GAR file**. This is done with *Ant.*
5. **Deploy service**. This is also done with *a GT4 tool*

3.2.2 Open -standards platform for grid computing

In the early days of grid computing, a number of custom middleware solutions were created to solve the grid problem, but this resulted in non-interoperable solutions and problematic integration among the participants. As stated earlier, the new wave of grid computing focuses on the easier integration, security, and QoS aspects of resource sharing. Foster described the Open Grid Services Architecture (OGSA) as a solution to the above problem. OGSA [8] provides a uniform way to describe grid services and define a common pattern of behavior for these services. It defines grid-service behavior, service description mechanisms, and protocol binding information by using Web services as the technology enabler. This architecture uses the best features from both the grid-computing community and the Web-services community.

3.2.3 OGSA architecture and goal

OGSA is a layered architecture, as shown in Figure 3.4, with clear separation of the functionalities at each layer. As seen in the figure, the core architecture layers are the Open Grid Services Infrastructure (OGSI) and OGSA platform services. The platform services establish a set of standard services including policy, logging, service level management, and other networking services. High-level applications and services use these lower-layer platform core components to become a part of a resource-sharing grid. The OGSA Grid services also address authorization, concurrency control, and manageability aspects.

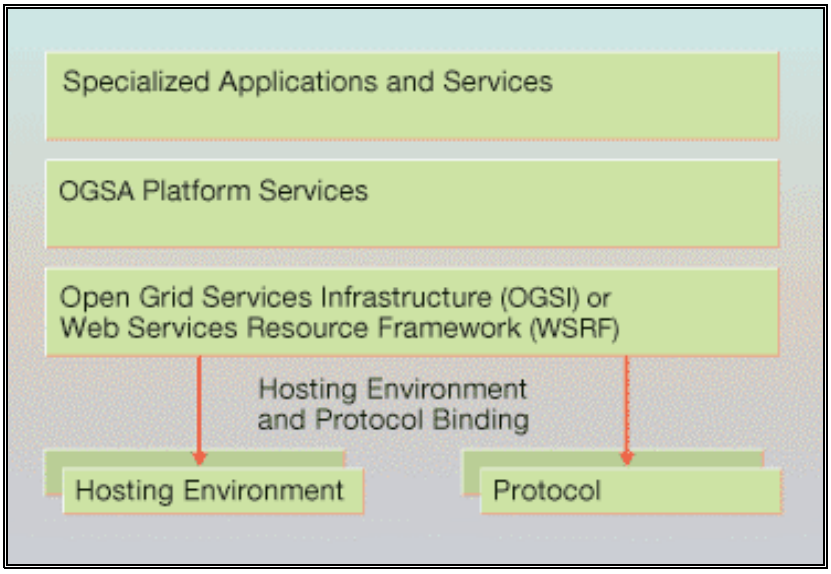


Figure 3.4 OGSA Architecture [8]

There are two standards currently available to implement OGSA-compliant Grid services:

- Open Grid Services Interface (OGSI) Grid services
- Web Services Resource Framework (WSRF) Grid services

Both frameworks provide mechanisms to implement OGSA-compliant Grid services in different ways.

3.2.4 Open Grid Services Interface (OGSI) Grid services

The Open Grid Services Interface defines rules about how OGSA can be implemented using Grid services that are Web services extensions. The OGSI specification defines a Grid service instance as "a Web service that conforms to a set of conventions expressed by WSDL as service interfaces, extensions, and behaviors."

The OGSI specification defines Grid services features that include:

- Statefulness
- Stateful interactions
- The ability to create new instances
- Service lifetime management
- Notification of state changes and Grid service groups

The OGSI model requires Grid services to be specified via Grid Web Services Definition Language (GWSDL), which is an extension of WSDL.

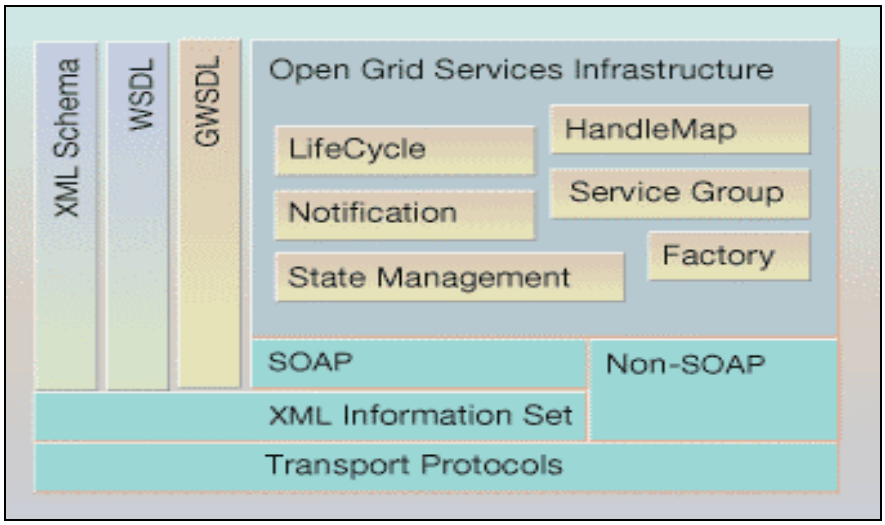


Figure 3.5 OGSI Components [8]

Figure 3.5 shows the layering of the OGSI components in a Web service with new interfaces and behaviors. The most notable point is the extension of WSDL to provide additional state data description mechanisms. In addition to this, the specification is a set of behaviors and interfaces to support service life-cycle stages: for example, service-level

management, collection management, state-change notifications, service creation mechanisms, and instance-reference management.

3.2.5 Web Services Resource Framework (WSRF)

WSRF [9] specification - an open framework for modeling and accessing stateful resources using Web Services, defines an approach to modeling, accessing, and managing state; to grouping services; and to expressing faults. It restates OGSI concepts in Web Services terms.

Specification	Description
Web-Services Resource Lifetime	It defines the mechanisms for WS-Resource destruction, including message exchanges that allow a requester to destroy a WS-Resource, either immediately or by using a time-based scheduled resource termination mechanism.
Web-Services Resource Properties	It defines how the type of definition of a Web Services resource can be associated with the interface description of a Web Services, and message exchanges for retrieving, changing, and deleting Web Services resource properties.
Web-Services Renewable References	It defines a conventional decoration of a WS-Addressing endpoint reference with policy information needed to retrieve an updated version of an end-point reference when it becomes invalid.
Web-Services Service Group	It defines a means by which Web Services and Web Services resources can be aggregated or grouped together for a domain specific purpose. In other words, it defines an interface to heterogeneous by-reference collections of Web Services. ServicesGroup can be used to form a wide variety of collections of services or resources, including registries of services and associated resources.
Web Services Base Faults	It defines an XML Schema type for base faults, along with rules for how this base fault type is used and extended by Web Services. In other words, it describes a base fault type used for reporting errors.
Web-Services Notification	The Web Services notification contains a family of specifications that defines a standard approaches to notification using a topic-based publish and subscribe pattern. The family of specifications comprises Web Services Base Notification (WS-BaseNotification),Web-Services BrokeredNotification (WS BrokeredNotification), and Web Services Topics (WS-Topics) [18].

Table 3.1 WSRF Specifications

The WSRF partitions OGSI into a family of composable specifications that define terms such as WS-Resource approach to modeling stateful resources with Web Services. It also

makes explicit distinction between the ”Service” and the stateful ”resources” acted upon by that service. Refactoring of OGSI yields five normative WS-Resource Framework and family of WS-Notification specifications.

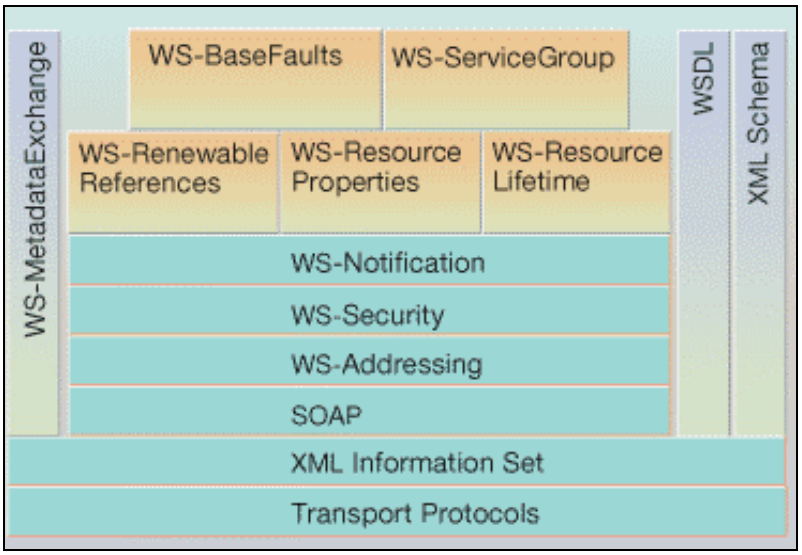


Figure 3.6 WSRF Specifications [8]

The mappings from the OGSI concepts and construct to equivalent WSRF concepts and constructs are presented in the following Table 3.2

OGSI	**WSRF**
Grid Service Reference	WS- Addressing Endpoint Reference
Grid Service Handle	WS- Addressing Endpoint Reference & WS- RenewableReferences
HandleResolver portType	WS- RenewableReferences
Service date definition	Resource properties definition
Grid Service portType	WS ResourceProperties & WS- ResourceLifetime
NotificationportTypes	WS- Notification
Factory portType	Factory Pattern
ServiceGroup portTypes	WS- ServiceGroup
Base fault type	WS-BaseFault

Table 3.2 Mapping from OGSI constructs to WSRF and WS-N Constructs

3.3 Relationship between OGSA, WSRF, and Web Services & GT4

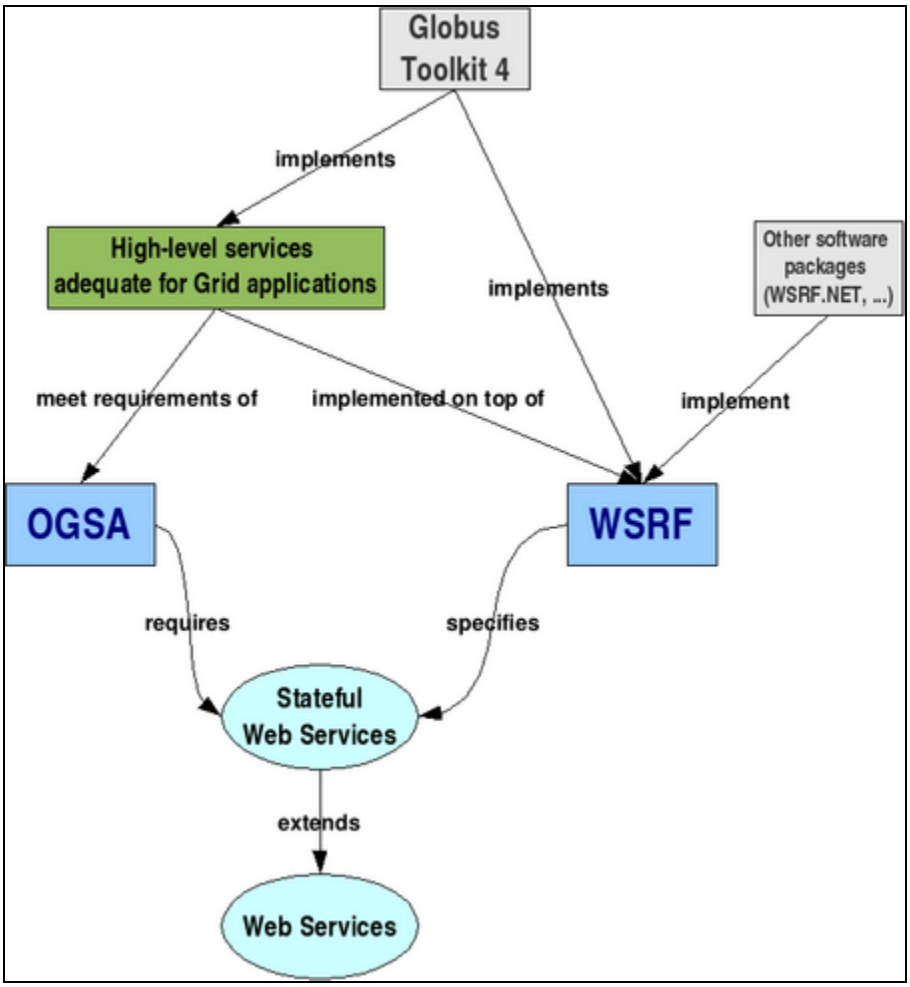


Figure 3.7 Relationship between OGSA, GT4, WSRF, and Web Services [22]

The GT4 toolkit, includes quite a few *high-level services* that are used to build Grid applications. These high-level services, in fact, meet most of the abstract requirements set forth in OGSA. In other words, the Globus Toolkit includes a resource monitoring and discovery service, a job submission infrastructure, a security infrastructure, and data management services , it *is* a realization of the OGSA requirements and a sort of *de facto* standard for the Grid community while GGF works on standardizing all the different services.

Most of these services are implemented *on top of WSRF* (the toolkit also includes some services that are not implemented on top of WSRF and are called the *non-WS components*). The Globus Toolkit 4, in fact, includes a complete implementation of the WSRF specification. This part of the toolkit (the WSRF implementation) is a very important part of the toolkit since nearly everything else is built on top of it. However, it's worth noting that it's also a *very small* part of the toolkit.

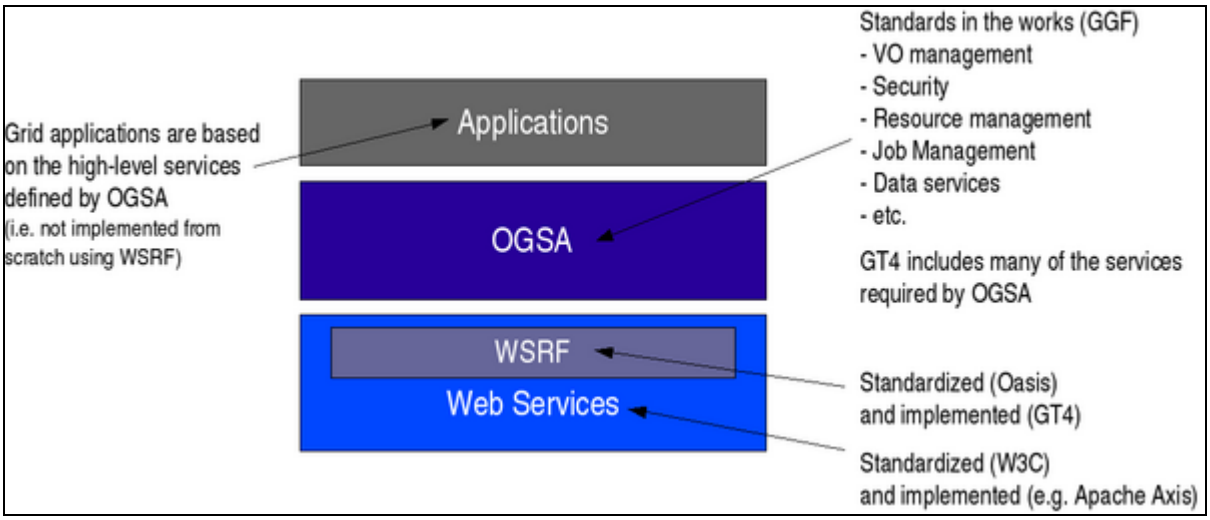


Figure 3.8 Layered diagram of OGSA, GT4, WSRF, and Web Services[22]

Chapter 4

Implementation Details and Experimental Results

4.1 Installation of GT4 on Linux

To establish the TIETGRID a basic requirement is the middleware. Our installation is on Red Hat Linux 9.0. Globus Toolkit is the well-known open source middleware for establishing Grid environment. The version we are going to use of this toolkit is GT4 [22] and it contains more new features than the older version of the Globus Toolkit. Following steps were carried out:

Step 1: Downloading GT4

Download GT4 toolkit from [22], source distribution is compatible with many UNIX-based operating systems.

Step 2: Preparing System effectively before Installation

Before starting the installation, prepare the system effectively as given below

- Check out the prerequisites and platform compatibility.
- Create a user named 'globus' in the local machine.
- Set the basic environment variable GLOBUS_LOCATION to the directory selected for installing GT4. In TIET GRID the GLOBUS_LOCATION= /usr/local/globus-4.0.1

Step3: Installing Prerequisites of GT4

Before going into installation of GT4, here is the list of requirements that are needed for installing grid middleware and the brief steps for their installation.

Prerequisites	Description	Installation Steps
Ant 1.6.5	Ant is a platform independent build tool required in both Windows and UNIX based platform before installing GT4. All the compilation of WSComponents is based on Ant. Junit package is needed along with Ant for some tests that may follow.	• Download the Binary Distribution of Ant • Extract it into the folder where to install it. • Set environment ANT_HOME to /usr/local1/ant Copy only the junit.jar from Junit distribution into Ant 'lib' Folder
J2SDk1.4.2.10	J2SDk is important for all the java based services.	1. Download J2SDk Binary for your platform and do the simple step of extract into the folder. 2. Set environment JAVA_HOME = /usr/local/j2sdk
Postgres 8.1.2	PostgreSQL is an open source implementation of RDBMS system supporting TCP/IP communication. You can just download and install the postgresql base distribution instead of downloading the full collection.	1 Set the variable PGDATA=/usr/local/pgsql/data 2 Change the owner to postgres user
Sudo 1.6.8 p12	sudo is used to run commands that require Super User privilege by the ordinary user itself. It is the important requirement for WS-GRAM. So, configuring sudo is very important for successful execution of jobs using WS-GRAM.	1. Download the binary. And Extract it into appropriate folder 2. set PATH if needed

Table 4.1 : Prerequisites for Globus installation

Step 4 Installation of GLOBUS

Set the variable GLOBUS_LOCATION=/usr/local/globus-4.0.1 and making the owner of this directory globus. Before getting into configuring of globus it is required that all paths are set correctly. Then following commands were executed.

globus$ cd /home/globus

globus$ tar –zxvf gt4.0.0-all-source-installer.tar.gz

globus$ ./configure –-prefix=$GLOBUS_LOCATION

globus$ make | tee build.log

root# make install

Step 5. Setting up Security

In a grid, each resource must have appropriate certificates to make it trust worthy. These certificates should be issued by an authority organization which is most trusted by the user and also the authority must know well about the user. This organization is in general called a Certificate Authority (CA).

- **Creating a new Certificate Authority (CA)**

The certificate authority is the incharge for issuing certificates for host and also the users of the grid. The certificate authority can be a most trusted external organization or even a personal certificate authority can be created using the software called *SimpleCA* provided along with Globus.

1. Run the script $GLOBUS_LOCATION/setup/globus/setup-simple-ca globus$ bash $GLOBUS_LOCATION/setup/globus/setup-simple-ca

2. **gt5.tiet.ac.in** is made to act as the **chief CA** host for the **TIET Grid.**

3. Subject of the CA installed in gt5.tiet.ac.in is **"cn=Globus Simple CA, ou=simpleCA, ou=gt5.tiet.ac.in, o=tiet grid"**.

4. Email of administrator of the SimpleCA at gt5.tiet.ac.in is **msingh@tiet.ac.in**. The clients can use this email to send the certificate requests.

5. Expiration date of the certificates issued by this CA is 5 years and it may change in future.

6. PEM pass phrase is the password or the key that is used to identify the CA. It is important since it has to be issued by the admin whenever he signs the certificates.

7. The CA hash is displayed at the summary of this setup. Note down the hash that displayed there which is used for configuring CA on other systems.

8. Final step is to setup Grid Security Infrastructure (GSI) for the grid.

globus$ cd $GLOBUS_LOCATION/setup/simpleca-<hash>/

globus$ bash setup-gsi –default

- **Obtaining required certificates from CA**

There are three types of certificates: Host Certificates, User Certificates, Proxy certificates.

Obtaining host certificate

Host certificate is obtained as follows:

1. Login as root,

 # grid-cert-request –host `hostname`

2. Three files are created in /etc/grid-security directory,

a. *hostcert.pem* – empty file to be replaced with certificate from CA.

b. *hostkey.pem* – holding the passphrase in RSA 1024-bit encrypted format

c. *hostcert_request.pem* – to be send to the CA for a signed certificate.

3. now, mail the hostcert_request to the Email of the CA.

4. **Signing of Host Certificates by CA:**

grid-ca-sign –in hostcert_request.pem –out hostsigned.pem

5. After that copy the signed cert hostsigned.pem into the /etc/grid-security/hostcert.pem.

Obtaining user certificate

User certificates are obtained as follows:

1. Every user needs a user certificate to run Globus Jobs. First, Login as the user.

2. Type, a. **grid-cert-request** – asks for your desired DN name and pass phrase. DN is nothing but the Distinguised Name similar to username in a local system.

3. It will leaves three files created in $(HOME)/.globus directory

a. *usercert.pem*

b. *userkey.pem*

c. *usercert_request.pem*

4. The steps are similar to that of the obtaining host certificates.

5. That is, mail the usercert_request to the CA. and get back the signed certificate and now copy it into $(HOME)/.globus/usercert.pem

Obtaining proxy certificate

Proxy certificates are at the lowest level of authentication process. It can also be used for testing whether the whole security is configured correctly. The command to do it as follows,

$ **grid-proxy-init**

gives output:

```
[globus@gt5 root]$ grid-proxy-init
Your identity: /O=Grid/OU=GlobusTest/OU=simpleCA-gt5.tiet.ac.in/OU=tiet.ac.in/CN=globus
Enter GRID pass phrase for this identity:
Creating proxy ........................... Done
Your proxy is valid until: Thu May 18 06:00:21 2006
```

The highlighted text is the DN of the user. DN is Distinguised Name that is used to identify a user in a grid. The success in this command means that the GSI is configured perfectly and the security is of no problem for this particular user.

Update grid-map file

The grid-mapfile is a file which contains the information of the various DN that can use that system as a grid resource, mapped with the local user name.

"/O=Grid/OU=GlobusTest/OU=simpleCA-gt5.tiet.ac.in/OU=tiet.ac.in/CN=globus"
globus

Creating container certificates and keys

Since only root has access to host certificates, some of the grid services need a separate host certificates for successful execution.

1. Login as root, and make a containercert.pem and containerkey.pem

root# cd /etc/grid-security
root# cp hostkey.pem containerkey.pem
root# cp hostcert.pem containercert.pem
root# chown globus:globus containerkey.pem containercert.pem

4.2 Implementing Simple Grid Service

Creating a simple MART service where item are put on sale and user can access this grid service in order to make their offerings for the item, a unique identity is created for the

item and the highest value of the offer placed on it is tracked therefore offer value must exist even if the user is not directly interacting with it, which makes this service "stateful". WSRF is a way to use "stateful" resources in the stateless environment of Web services. WSRF is about changing that environment into one in which you can create a persistent resource and manage it through a Web service. That combination is called a WS-Resource, we require WS-core files for using the WSRF. There is a Grid service that interacts with the user enabling us to place our offerings. The combination of the service is the WS-Resource.

File	Description
Mart.java	It is the resource. This class actually manages all the actions we're going to take with regard to the offers being made for the item.
MartHome.java	It is the ResourceHome. This class is responsible for creating multiple instances of the resource and for helping locate a specific instance, as requested.
MartService.java	It is the service that will interact with the resource Mart.java. This class contains methods which then call the resource class to actually do the work.
build.xml	Containing the instructions that guide the entire build process.
Deploy-server.wsdd	This file explains how to link the service with the actual class that implements it. This file is of utmost importance as we will set security parameters in this file itself.
NStoPkg.properties	This file specifies how to derive the package names from the namespaces if they need to be different from what can be derived from the service's WSDL file.
post-deploy.xml	Provides instructions for miscellaneous instructions to be carried out after Ant finishes the deployment.
deploy-jndi-config.xml	This file links the service, the resource, and the ResourceHome so the server knows where to look when it receives a request.

Table 5.2: List of files of the Mart Service

Following is the list of actions that can be performed by the client.

- *startoffer* for the item available at Mart, this action generates a new unique number associated with the item which will persist till it is destroyed using delete action, here initial value of the offer for this unique item number is 0.
- *myoffer*, this action when performed takes value from the user of his offer, this value will be written to the unique item id, and is updated if and only if a higher value is offered.
- *delete* the item, this action allows to delete an item on the mart .

Our main task is to create a Mart Service and then incorporate the security feature into it for that matter we will be writing Security Descriptor and Security Configuration file.

First creating the grid service without security feature, this service will be available to all the user who have access to grid machines because security is not implemented into this service which is a very obvious breach to security of the grid service. In this case our security descriptor file deploy-server.wsdd for simple MartService will look like this

```
<deployment name="defaultServerConfig" xmlns="http://xml.apache.org/axis/wsdd/"
xmlns:aggr="http://mds.globus.org/aggregator/types" xmlns:java=
"http://xml.apache.org/axis/wsdd/providers/java" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<service name="MartService" provider="Handler" use="literal" style="document">
<parameter name="providers" value="GetRPProvider"/>
<parameter name="handlerClass" value= "org.globus.axis.providers.RPCProvider"/>
<parameter name="scope" value="Application"/>
<parameter name="allowedMethods" value="*"/>
<parameter name="activateOnStartup" value="true"/>
<parameter name="className" value= "org.globus.tutorial.mart.MartService"/>
<wsdlFile>
share/schema/tutorial/mart_service.wsdl
</wsdlFile>
</service>
</deployment>
```

Following steps were carried out for simple MartService:

Step 1.Start a terminal window, we will call this window W1, and set the value of GLOBUS_LOCATION to /us/local/globus-4.0.1

Step 2. Set ANT_HOME to /usr/local/globus-4.0.1/ant

Step 3. Set JAVA_HOME to / usr/local/globus-4.0.1/j2sdk

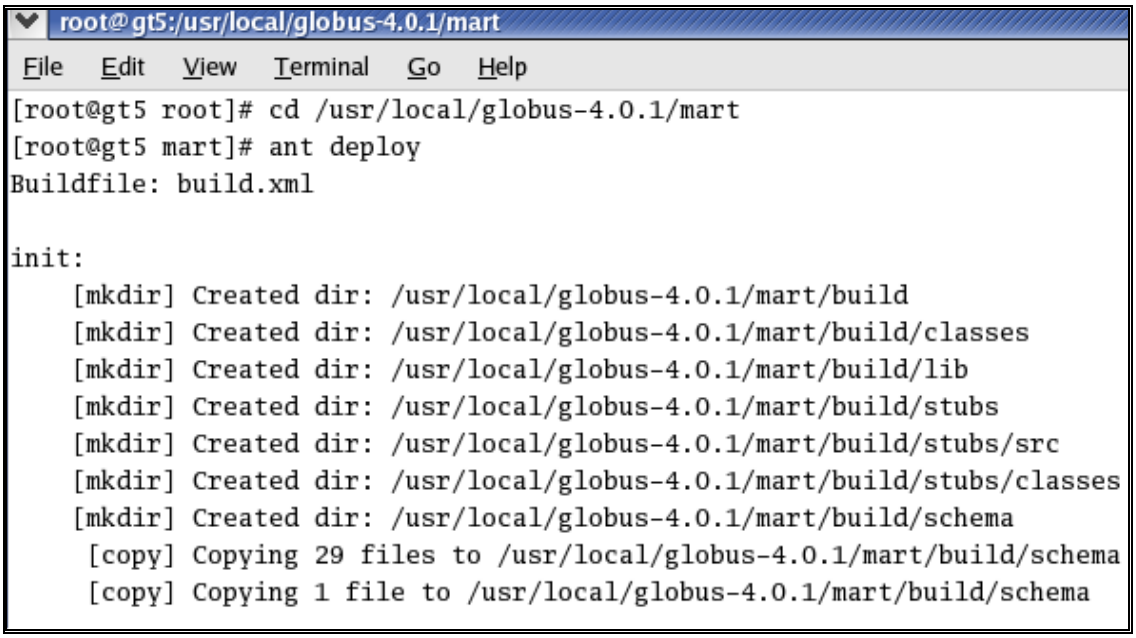
Step 4. Make sure JAVA_HOME/bin, ANT_HOME/bin and GLOBUS_HOME/bin are on the path, as in:

PATH=$GLOBUS_LOCATION/bin:$JAVA_HOME/bin:$ANT_HOME/bin:$PATH

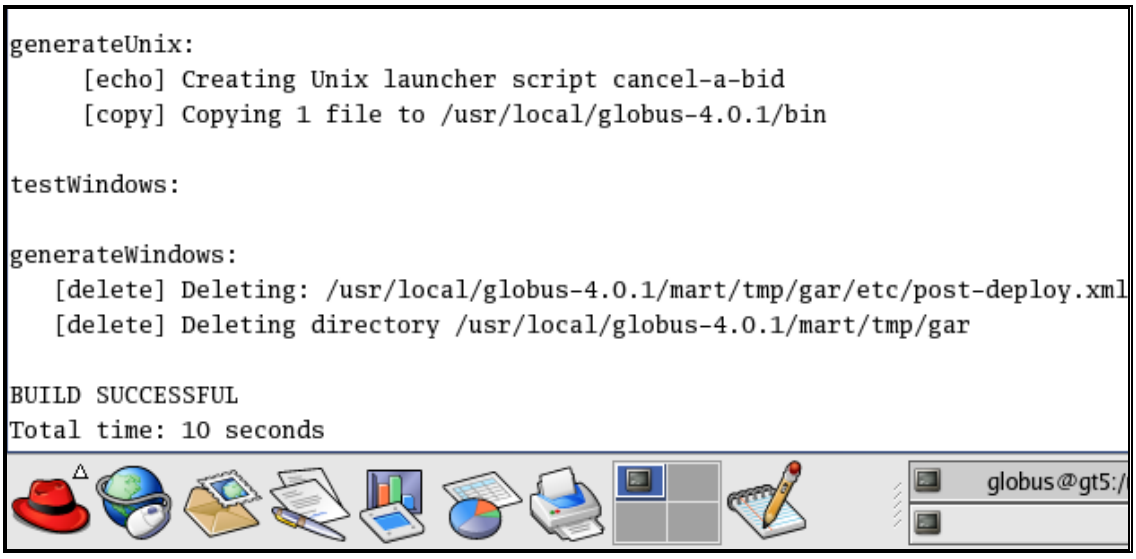
export PATH=$PATH

Now navigate into the home directory of the MartService that is mart and from there do the following a steps.

Step 5. Deploy the service using ant tool



Screen Shot 4.1: Deploying the simple Grid Service



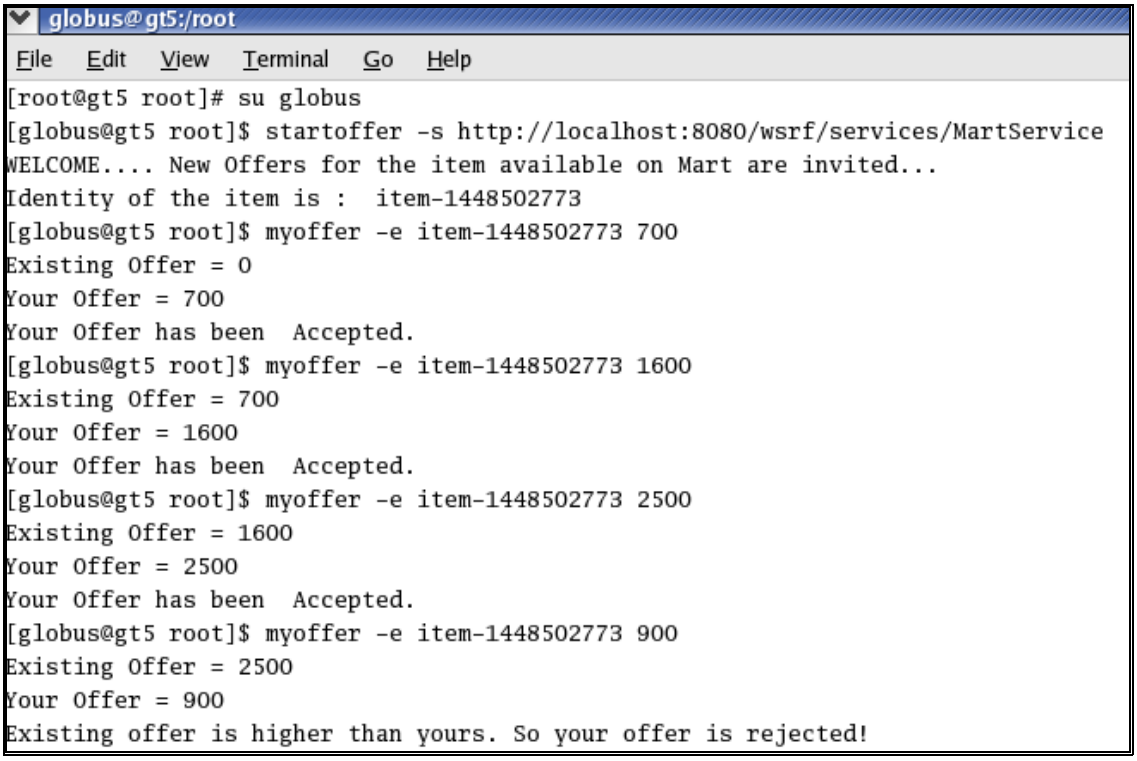
Screen Shot 4.2: Build Successful for simple Grid service

Step 6. If the build is successful then open a new terminal window, we will call this terminal window W2 and start the globus container without security using –nosec option following will appear in this window.

Screen Shot 4.3 : Globus Container without security

As a result we will see the service will be deployed into the container, here our MartService appears as service number 13. All the services are at Port: 8080 because the container has been started with –nosec option.

Step 7. Now in terminal window W1 you can perform the allowed user actions as discussed earlier, following will appear on the terminal window



Screen Shot 4.4: Accessing the Grid Service without authentication

This can be observed that since no security criteria was considered in this implementation of the service so service is available to any user who have access to the grid machine without considering authentication need for the service.

4.3 Implementing Transport Level Security in Grid Service

Now will implement the MartService grid service with service level security being incorporated into it , our main aim is to implement the service with transport level security and check for the same by experimental results.

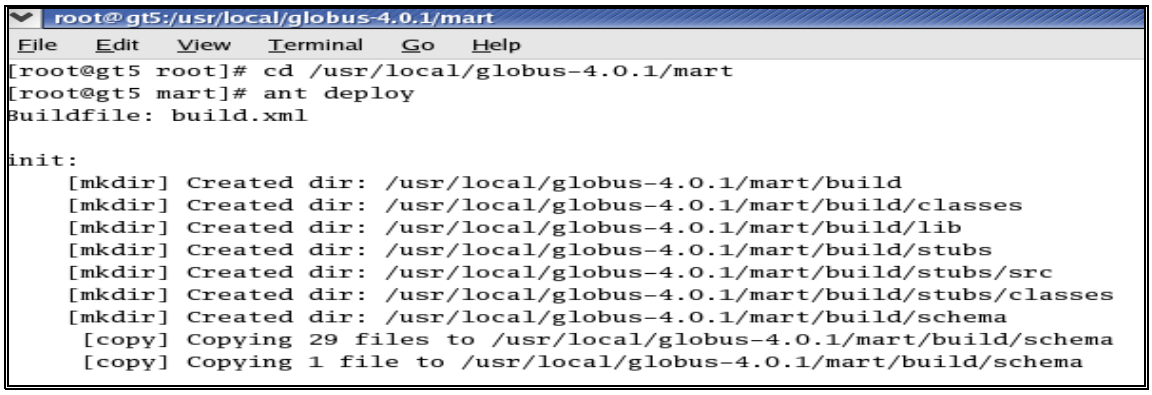
In this case our security descriptor file **deploy-server**.wsdd looks like this

<deployment name="defaultServerConfig" xmlns="http://xml.apache.org/axis/wsdd/"
xmlns:aggr="http://mds.globus.org/aggregator/types" xmlns:java=
"http://xml.apache.org/axis/wsdd/providers/java" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<service name="MartService" provider="Handler" use="literal" style="document">
<parameter name="providers" value="GetRPProvider"/>
<parameter name="securityDescriptor" value=/etc/globus_tutorial_mart/security-config.xml>
<parameter name="handlerClass" value= "org.globus.axis.providers.RPCProvider"/>
<parameter name="scope" value="Application"/>
<parameter name="allowedMethods" value=""/>*
<parameter name="activateOnStartup" value="true"/>
<parameter name="className" value= "org.globus.tutorial.mart.MartService"/>
<wsdlFile>
share/schema/tutorial/mart_service.wsdl
</wsdlFile>
</service>
</deployment>

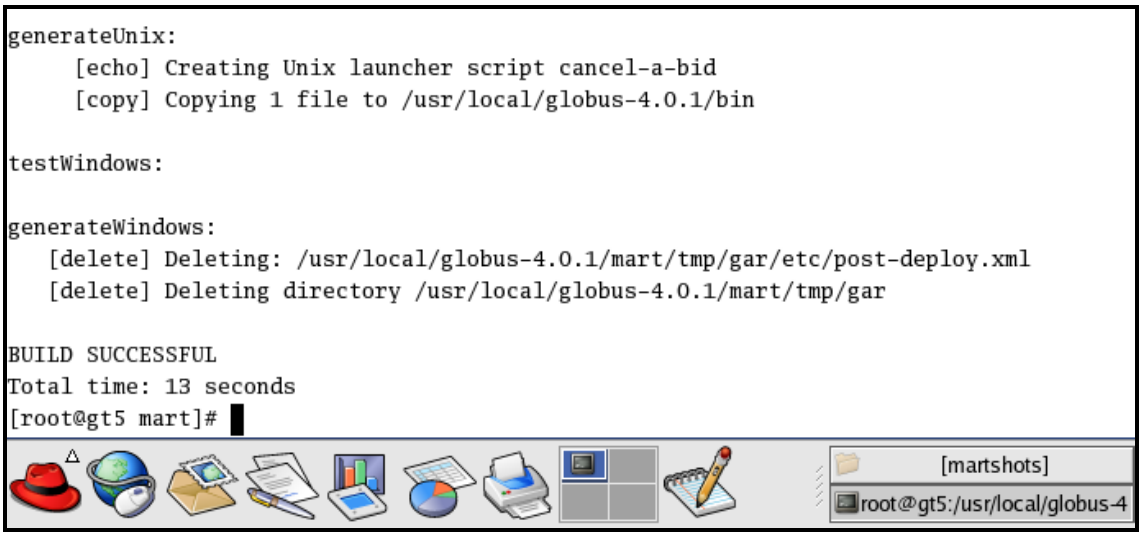
Before deploying the Secure MartService, first we need to undeploy the simple MartService that we had deployed earlier in terminal window W1, in order to do that navigate to the home directory for the MartService that is /usr/local/globus-4.0.1/mart and type **ant undeploy.**

Following steps were carried out for Secure MartService:

Step 1. Deploy the service using ant tool.



Screen Shot 4.5: Deploying Secure Grid Service



Screen Shot 4.6: Build successful in secure grid service

Step 2. Once service is deployed , write a file called **security-config.xml** and save it in **/usr/local/globus-4.0.1/etc/globus_tutorial_mart/ security-config.xml**. This is the file where we set the **TRANSPORT LEVEL SECURITY**. The content of the file is

```
<securityConfig xmlns="http://www.globus.org">
<authz value="none"/>
<!-- Default for all methods -->
<auth-method>
  <GSITransport>
    <protection-level>
      <integrity/>
    </protection-level>
  </GSITransport>
</auth-method>
</securityConfig>
```

Step 3. If the build is successful the open a new terminal window W2, and start the globus container with security following will appear in this window.

As a result you will see the service will be deployed into the container, here our MartService appears as service number 13.All the services are at Port:8443 because the container has been started with security.

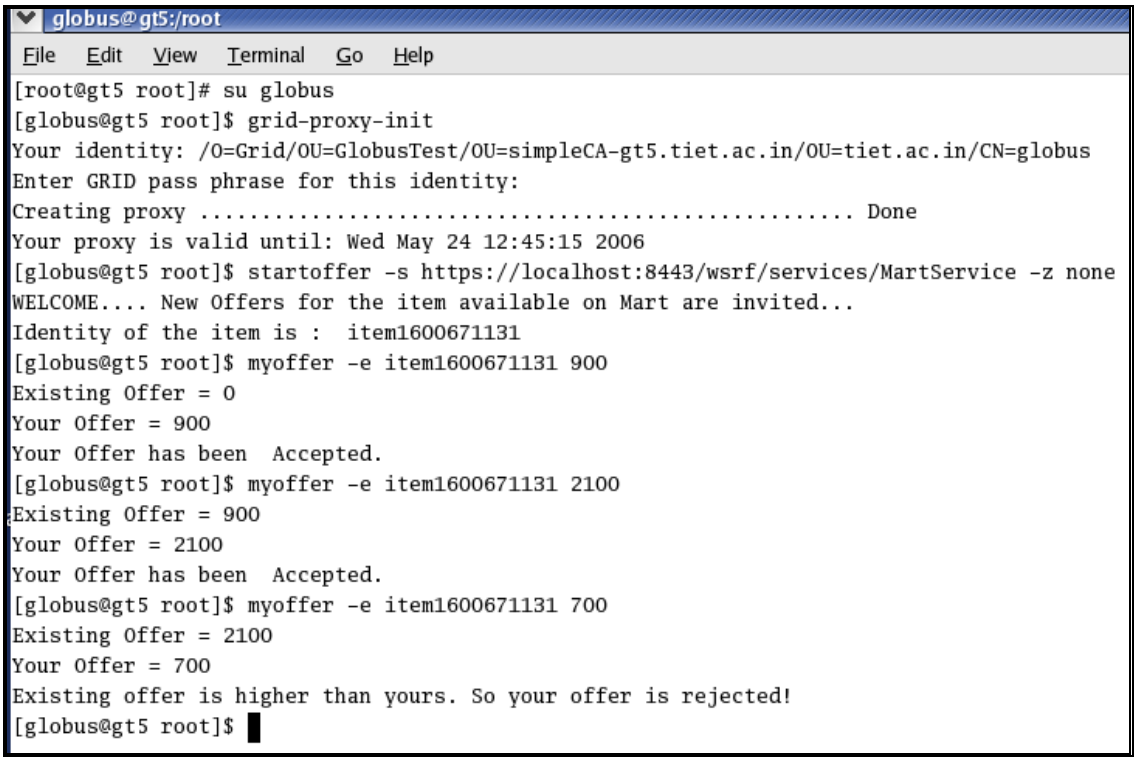
Screen shot 4.7: Globus container with security

Step 3. Now in terminal window W1 you can perform the allowed user actions as discussed earlier, we will generate the proxy certificate of a user who is allowed to use the grid and have X.509 certificates in the /etc/grid-map-file, then we will access the MartService with authentication verification on the basis of

- Generating valid user proxy certificates and accessing the MartService.
- Destroying the user proxy certificates and trying accessing MartService.
- Regenerating the user proxy certificates and continuing with the previous item on MartService.

4.4 Experimental Result 1

User is a certified one and his proxy certificates are generated using **grid-proxy-init** and once the proxy certificates are generated user will be authenticated based on that , if the user is the certified one then he/ she will be allowed to perform the actions



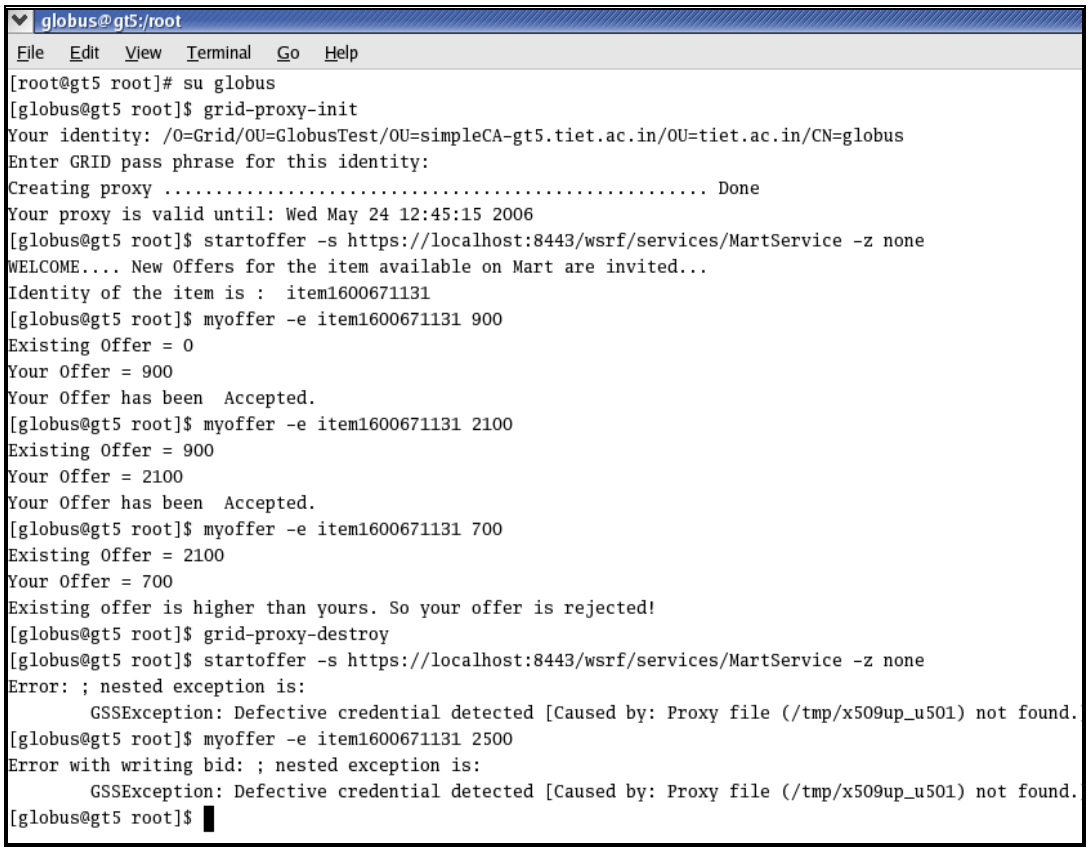
Screen shot 4.8: Experimental result 1

Explanation:

Here first of all proxy certificates for the user are created using grid-proxy-init, and then user tries to access the service by starting offer for the item on Mart. Because the user carries the appropriate credentials therefore user is allowed to perform the desired action, user places his offer for the item number 16006731with a value 900, since initial value of the offer for this item is 0, so the action is accepted and offer value for the item is updated to 900, then follows the offer of 2100, then finally 700, since this offer is lesser than previous offer of 2100 so offer of 700 is rejected. This screen shot shows that if the user got the valid proxies than only he is offered service otherwise not.

4.5 Experimental Result 2

User proxy certificate are destroyed using **grid-proxy-destroy**, then user tries to perform the allowed actions, as per the security setting it should result in error



Screen shot 4.9: Experimental result 2

Explanation:

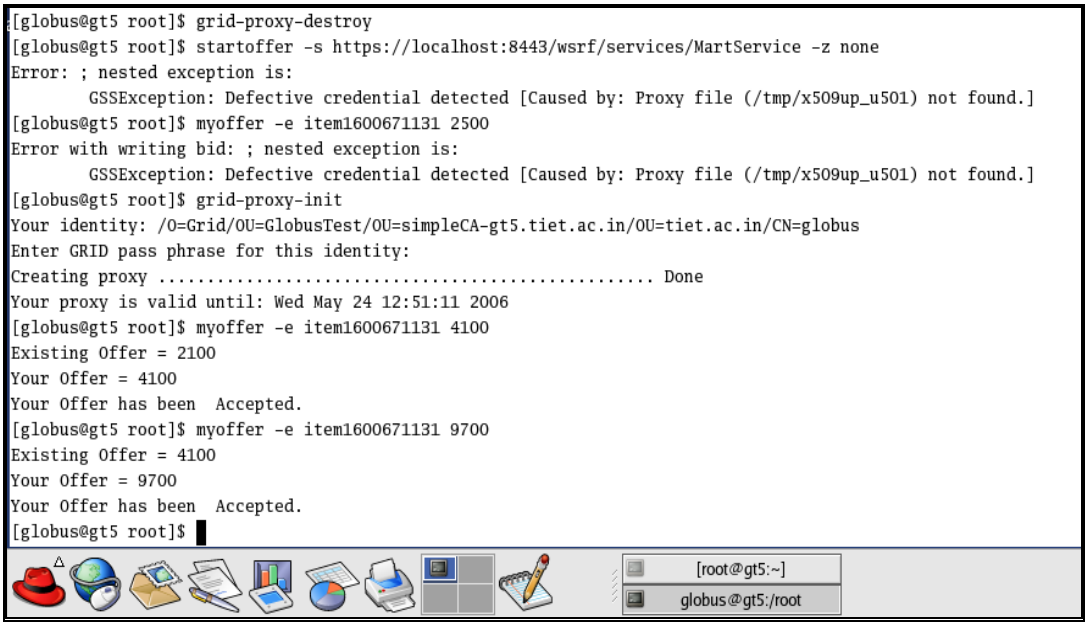
Here proxy certificate of the user are destroyed using grid-proxy-destroy, then user without proxy is made to access the service. As valid proxy certificate are not provided so the authentication fails and service is denied. An error occurs that states

GSSException: Defective credential detected [caused by :Proxy file (/tmp/x509_up_u501) not found.]

This basically shows that once the user looses his proxy certificates, he is no more allowed to access the service as the authentication mechanism fails.

4.6 Experimental Result 3

After Destroying the user proxy certificates again proxy certificates are generated and the same item number is accessed for some allowed action, the service should again be accessible with value of the previously performed successful action.



Screen shot 4.10: Experimental result 3

Explanation:

Here once again the proxy certificates are generated and again user access the same service and place an offer worth 4100, which is greater than the persisting offer worth 2100, since valid credentials are again with the user, so authentication succeeds and action is also successful this proves the statefulness of the service as well.

Chapter 5

Conclusion and Future work

5.1 Conclusion

Traditionally, security has been concerned with authenticating and authorizing the users and maintaining the confidentiality, integrity and availability of data resources. On the Grid, resources involved may be extremely sensitive and valuable. This requires that only authorized project users can gain access to read or modify these resources for that purpose identity of the user accessing the resources need to be verified through some authentication mechanism. For that matter a simple authentication infrastructure is needed so that users and owners should be protected from each other. Resources availability is vital on the Grid because without it the purpose of sharing resources will not be accomplished. The smooth running of the Grid project crucially relies on security mechanisms that ensure these attributes.

The potential benefits of the Grid are enormous but the biggest barrier for their wider adoption is security. Currently a limited concept of virtual organization is realized by the Grid for scientific collaborations but the domains of potential applications is huge: VO possibilities range from e-government, health, e-learning to military and the coordination of multinational forces. Grid security in GT 4, has made a strong leap forward in recent years in many aspects- (for example authentication, confidentiality). The innovative applications of cryptography, PKI, and X509 have been the source of many of the radical advances in the evolution of security solutions to these aspects.

The distributed services provide promising new tools and technologies in distributed computing environment. The convergence of Web Services and Grid Services into WSRF provide flexible solutions to the distributed service providers by enabling stateful Web Services. Finally, the implementation of Grid Services by implementing TRANSPORT LEVEL SECURITY has been presented.

5.2 Future Work

The procedure used by the Grid to authenticate candidate users must be strong in order to avoid identity theft at the point of certificate creation. CA computer systems and applications must be protected physically and protected from tampering according to best security practices. Public key of the CA must be securely communicated to users and VO sites.

In this thesis we have implemented a simple and then a secure grid service in order to implement the TRANSPORT LEVEL SECURITY in grid environment. Grid service with limited functionality is developed so in future work a typical secure grid service with huge complex functionality can be developed by following the same steps.

Here the environment used was GT4 on red Hat Linux 9.0, so same experiments can be further carried out in the heterogeneous environment, say cross platform, for example the web service runs on a windows machine and user tries to access it from a Linux machine.

References

[1] D. Gollman. "Computer Security". John Wiley, 1999.

[2] Foster, C. Kesselman, and S. Tuecke, *The Anatomy of the Grid--Enabling Scalable Virtual Organizations*, The Globus Alliance, http://www.globus.org/research/papers/anatomy.pdf.

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

[4] I. Foster, C. Kesselman, G. Tsudik and S. Tuecke "*A Security Architecture for Computational Grids*". *ACM Conference on Computers and Security*, 1998,83-91.

[5] I. Foster, L. Pearlman, V. Welch, C. Kesselman, S. Tueke "*A Community Authorization Service for Group Collaboration*",http:// www.Globus.org

[6] IBM Grid solutions: http://www-1.ibm.com/grid/solutions/index.shtml

[7] J. Joseph, C. Fellenstein, "Grid Computing", LPE, Person Education, 2004.

[8] Jacob, Brown, Fukui and Trivedi " Introduction to Grid Computing", First Edition, Dec 2005,http://www.redbooks.ibm.com/

[9] K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, T. Maguire, D. Snelling, and S. Tuecke, *From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring & Evolution* (March 2004), http://www-106.ibm.com/ developerworks/library/ws-resource/ogsi_to_wsrf_1.0.pdf.

[10] M. Baker, R. Buyya and D. Laforenza "Grids and Grid technologies for wide-area distributed computing", Software Practice & Experience. 2002

[11] M. Surridge "*A rough Guide to Grid Security*". Issue 1.1a, IT-Innovation centre, 2002.

[12] R. Buttler, V. Welch, D. Engert, I. Foster, S. Tuecke, J. Volmer, C. Kesselman "*A National-Scale Authentication Infrastructure*", IEEE, Dec. 2000

[13] Sotomayor, "The Globus Toolkit 4 Programmers Tutorial", 2005, http://www.gdp.globus.org

[14] UK E-Science programme: http://www.research-councils.ac.uk/escience/

[15] V. Welch, F. Siebenlist, K.Czakowski, J.Gawo, S. Meder, L. Pearlman, S. Tueke "Security for grid Services",http:// www.globus.org

[13] V. Welch, Ian Foster, "X.509 Proxy Certificates for Dynamic Delegation", IEEE, 2004.

[17] William Stallings, "Cryptography and Network Security Principles and Practices", Third Edition, Prentice Hall, 2003.

[18] World wide web consortium (W3C): Leading the web to its full potential. Online. http://www.w3c.org

[19] www.entrust.com

[20] www.gdp.globus.org

[21] www.ggf.org

[22] www.Globus.org

[23] www.gridcomputing.com

[24] www.ibm.com

[25] www.microsoft.com/windows

[26] www.rsa.com

[27] www.unix.org

[28] www.verisign.com

List of Publications

1. Jagdish Kumar, Dr. Seema Bawa and Maninder Singh , "Authentication in Grid Environment ", in Proceedings of National Conference on Recent Trends In Engineering And Computational Techniques (REACT-2006), BGIET Sangrur , 5-6 April,2006. *(Published)*

2. Jagdish Kumar, Dr. Seema Bawa and Maninder Singh, " Implementing transport level security in grid services using GT4 ", International symposium on Parallel & Distributed Processing & Applications (ISPA' 2006), DII Second University of Naples, Sorrento, Italy, 1-4 Dec 2006.