Security development for the trustworthy evolving grid

Zhu, Ning

2006

[illegible]

SECURITY DEVELOPMENT FOR THE TRUSTWORTHY EVOLVING GRID

ZHU NING

SCHOOL OF COMPUTER ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY

2006

QA
76.9
.C58
Z63
2006

Security Development for the Trustworthy Evolving Grid

Zhu Ning

School of Computer Engineering

A thesis submitted to the Nanyang Technological University in fulfilment of the requirement for the degree of Master of Engineering

2006

ACKNOWLEDGEMENTS

I am indebted to many people who contributed to this research. I would like to give my cordial thanks to them here.

First and foremost, I owe the greatest gratitude to my supervisor Dr. Stephen John Turner and co-supervisor Dr. Jussipekka Leiwo, for their guidance, heuristic advice and encouragement in my research. Dr. Turner and Dr. Leiwo constantly support my work and give me insightful suggestions, and they have made enormous efforts in this research. I really appreciate their rigorous spirit in research and benefit a lot from their numerous experiences and great geniuses. Without their support, I could not have gone so far in my research work.

I would also thank Dr. Cai Wentong, Dr. Lee Bu Sung, and Dr. Bertil Schmidt for their patience in listening and discussing with me in the group meetings. Their valuable counsel always inspires me. I also appreciate Mr. Li Yikun, Mr. Tang Ming, Mr. Wang Lizhe, Ms. Feng Yuhong, Mr. Xu Xiang, Mr. Xavier Percival, Mr. Ho Quoc Thuan, and all others who have provided valuable advice and patient help for me. They all contributed to my research work in one way or another.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

AC – X.509 Attribute Certificate

ADF - Access control Decision Function

AEF - Access control Enforcement Function

BP - Base Practices

CA - Certificate Authority

CC - ISO/IEC standard 15408, the Common Criteria

EAL - Evaluation Assurance Level

GP - Generic Practices

GSI - Grid Security Infrastructure, the security component of Globus Toolkit

GT - Globus Toolkit

ICT - Information and Communications Technology

ITSEC - Information Technology Security Evaluation Criteria

LDAP - Lightweight Directory Access Protocol

OGSA - Open Grid Services Architecture

OGSI - Open Grid Services Infrastructure

PA - Process Area

PKI - Public Key Infrastructure

PP - Protection Profile

SAML - Security Assertion Markup Language

SFP - Security Functional Policy

SOA - Source of Authority

SSE-CMM - Systems Security Engineering - Capability Maturity Model

SSL - Secure Sockets Layer

ST - Security Target

TCSEC - Trusted Computer System Evaluation Criteria

TOE - Target of Evaluation

TSF - TOE Security Functions

TSP - TOE Security Policy

UML - Unified Modeling Language

VO - Virtual Organization

WSRF - WS-Resource Framework

XACML - eXtensible Access Control Markup Language

ABSTRACT

Grid computing is concerned with the sharing and coordinated use of distributed diverse resources among virtual organizations. Its distinctive features bring challenging security issues that require new approaches to trustworthy specification, design, implementation, test, deployment and operation in the Grid security development.

This project makes a novel contribution to Grid security by addressing security assurance issues and applying a disciplined security engineering process for the development of high-assurance Grids. In order to realize security functions and architectures in an assured manner for the evolving Grid, an effective method is to analyze Grid security at a high level, derive disciplined security objectives and requirements, and perform security design, implementation and test following a recognized assurance framework. The generic security specifications summarize core Grid security artifacts and their dependencies, and constitute the foundation of the Grid security assurance framework proposed in this thesis which can be used to guide the development and evaluation of trustworthy Grids.

A security environment has been systematically defined and security objectives derived for a generic Grid-based application execution and information sharing engine in this thesis. The security functional and assurance requirements are specified, tailored, and grouped to satisfy the objectives. Then an associated specific lower level security design and trustworthy implementation following the proposed assurance framework are presented. Such Grid security implementations are disciplined instantiations of generic security specifications, instead of simply being developed proprietarily in an undisciplined way. Their functionalities are highly assured by enforcing various assurance

measures to conform to the specific system/application technical needs and meet the security objectives correctly.

Chapter 1

Introduction

1.1 Overview of the Problem

The Grid, a distributed computing infrastructure for advanced science and engineering, has been developing very fast to be one of the most attractive and active IT areas, and the achievements of current Grid implementations have positively demonstrated the power and potential of Grid computing. The main distinction of the Grid from conventional distributed systems is that it focuses on large-scale distributed heterogeneous resource sharing and seamless access for high performance innovative applications [FKT01]. Many people think the Grid will grow into the "next generation Web". In fact, where the Web gives us access to a wide range of raw information, the Grid can further provide us with access to computing facilities and allow us to process the information to get the answers we really need. There are two major benefits of Grid computing: savings and speed, and the benefits can also be divided into business benefits and technology benefits [GEL04]. As more and more institutions and people from business and industry are involved with the aim of improving and expanding their transactions and abilities, there even appear demands for more comprehensive, powerful and versatile infrastructures and techniques for the new visions of the Grid than those for current academic oriented Grids.

For the Grid, security has a more imperative meaning because Grid resources are often expensive hardware/software, applications and services that usually belong to important science, engineering, business, or defense institutions, and users may run their large and crucial jobs which should be protected during their communications and executions on the Grid. Therefore, damage to facilities, information and programs in the Grid can be much more severe than those in traditional distributed environments and security has to be addressed in depth as a key issue in Grid research and development. However, as shown in subsection 1.3.2 and section 2.1, many current Grid security solutions may not be as secure as they claim and do not properly fulfill their security objectives because of their non-systematic development processes. Any error, mischance, or inconsistency occurring in or among its casual or ad hoc specification, design, implementation, test, and/or other life cycle stages can make a Grid security system untrustworthy even though it was built on recognized security techniques. Simply embedding security considerations in the architectural design and coding is not enough and we need security engineering approaches to assure reliable Grid security functionality. These have not yet been addressed by the Grid community.

In trustworthy development, effective and efficient methods of systematically modeling, specifying, implementing and evaluating Grid security need to be followed to build highly assured systems/applications for the Grid in a disciplined way. The fulfillment of a full set of systematically derived and expressed Grid security artifacts and specifications with consistent design and high quality security techniques will bring a dependable, manageable, and more powerful novel distributed computing and application platform to be enjoyed widely in the future.

The contributions of the work summarized in this thesis include the following:

- Addressing security assurance issues for the development of trustworthy Grids, which was not covered by previous Grid security efforts;
- Deriving, refining, and justifying a comprehensive suite of generic security specifications for the Grid which are disciplined using internationally recognized IT security standards and assurance methods and can be used as the basis and reference of systematic Grid security development;
- Proposing a Grid security assurance framework for a security engineering approach to flexibly develop trustworthy Grid security architectures and implementations; and
- Carrying out a specific Grid security development following the Grid security assurance framework and security engineering approach to gain technically improved and highly assured security functionality in the final implementation.

1.2 Grid Computing

With the rapid development of powerful computers and high speed networks, it has been possible for us to use distributed computers as a single unified computing resource, which led to the definition of the *computational Grid* [FK98] in 1998 - *A hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high end computational capabilities.* A computational Grid focuses on providing raw computing power, high speed bandwidth, and associated data storage in a secure and auditable way.

As a result of the advancement of Grid research, in 2001 Ian Foster and Carl Kesselman gave a relatively less technical definition for the Grid [FKT01] - *flexible, secure, coordinated resource sharing and problem solving among dynamic collections of* *individuals, institutions, and resources.* A set of individuals, institutions and/or resources defined by proper sharing rules is called a Virtual Organization (VO) herein.

Although most Grid projects have implemented the interoperability which is essential to achieving large-scale and intensive high performance computation, it is also widely desirable to be able to reuse existing components and assemble them in a flexible manner for building new Grid systems/applications. The possible solutions require an architectural evolution where the adoption of a service-oriented model and the attention to metadata increase. The Globus [GLOBUS] team proposed a new service-oriented vision for the Grid: the Open Grid Services Architecture (OGSA) [FKN02] in 2002. It represents a long-overdue effort to define an "architecture" for the Grid [GCG02], and converges Web Services [WS] and Grid computing to support the creation, maintenance, and application of ensembles of services maintained by VOs. In Web Services, WSDL (Web Services Description Language) [WSDL] is used to describe how to interface with a service rather than the functionality of that service, SOAP (Simple Object Access Protocol) [SOAP] is used to encapsulate messages between services, and UDDI (Universal Description, Discovery and Integration) [UDDI] is used for publishing, finding, and binding registered services. XML and XML Schema are also used to form layers to separate the concerns of transfer, description, and discovery. Grid Services defined by the OGSA considerably extend Web Services for dynamic and volatile, ad-hoc, large, or long-lived configurations. To achieve the flexible assembly of Grid components and resources, information about the functionality, availability and interfaces of various components is needed. It is embodied by the metadata and must have an agreed interpretation. Because service discovery is through the metadata descriptions and service composition is also controlled and supported by them, OGSA addresses metadata issues which have become critical to achieving a successful service based Grid environment [RJS02]. Core features of the OGSA are specified in OGSI (Open Grid Services Infrastructure) [OGSI] for implementation and were characterized by Globus Toolkit (GT) 3 [SG03]. In 2004, a refactored and evolved version of OGSI, the WSRF (WS-Resource Framework) [WSRF], was further brought out to reflect some functional improvement and coordinate with the Web Services community better. The emerging Globus Toolkit 4 [FOS05] is the first implementation of WSRF.

In brief, the four main aspects that characterize a Grid are [BBL02]: 1) Multiple administrative domains and autonomy; 2) Heterogeneity; 3) Scalability; and 4) Dynamicity or adaptability. The Grid is currently mainly a research infrastructure used by scientists and engineers who want to share their computers to run ever larger calculations, but it is already entering business and industry. Several typical projects and techniques for the Grid are shown below:

- Globus [GLOBUS] - It is a multi-institutional research effort that seeks to enable the construction of computational Grids which provide high end computing power, and deals with the way that computational resources are allocated, scheduled, and executed.
- Legion [GRI97] – It provides a software infrastructure by which a system of heterogeneous and geographically distributed high performance computing machines could implement seamless interactions among the resources involved.
- SRB [RM01] - It has been developed to provide "uniform access to distributed storage" across a range of storage devices via a well-defined API.
- Nimrod/G [BAG00] [AGK00] and GRACE [BGA00] - Nimrod/G is a Grid broker that performs resource management and scheduling of parameter sweep and task-farming applications. It has been adopted in the GRACE project to support user-

defined deadline and budget constraints for scheduling optimizations and to manage the supply and demand of resources in the Grid.

- The SDSC Grid Port Toolkit [SDSC] – This is a reusable portal toolkit that builds on NPACI Hotpage [HOTPAGE] infrastructure.
- Cactus [ADF01] – It is an open source problem-solving environment designed for scientists and engineers.
- DataGrid [DG] - The DataGrid project aims at setting up a computational and data-intensive Grid of resources for the analysis of data coming from scientific explorations [HJS00].
- Access Grid [ACGRID] – It is a tool for supporting group-to-group interactions across the Grid, and consists of a collection of resources that support visual human collaboration such as large-scale distributed meetings and training.

It has also been noted that there is no mature Grid middleware that provides an open system with a high degree of easy to use and seamless automation for flexible collaborations and computations on a global scale [RJS01]. A possible solution to this deficiency is provided by the Semantic Web [SEMAN]. The idea of the Semantic Web is to have data on the Web defined and linked in a way that allows more effective discovery, automation, integration, and reuse across various applications. Consequently, the utilization of Semantic Web technologies to build a new generation Grid infrastructure will lead to the Semantic Grid [GR02] [RJS02], and the two experimental projects are myGrid [MYGRID] and CoAKTinG [SRE02]. In the Semantic Grid, data, information, knowledge and services are given well defined meaning to better enable computers, automated tools and people to work in cooperation.

1.3 Motivation of the Research

1.3.1 Security – Functionality vs. Assurance

From the security engineering point of view, security should be expressed as both functionality and assurance. However, people often simply characterize computer security as a set of functional components implemented in products and systems. The security functional components directly protect the system and fulfill security requirements which are intended to counter threats to assets in the assumed operating environment and cover any identified organizational security policies and assumptions. Security functionality and technique relevant issues have been a main research area of the IT security community, and a system or product will be claimed secure if it implements recognized security techniques, for example, known cryptographic algorithms. Unfortunately, the strengths of underlying security techniques cannot be translated directly into statements about the strengths of mechanisms implemented in systems and products. Most practical attacks exploit weaknesses in the design and implementation of security functional components, not in the properties of the underlying techniques. The weaknesses were brought about by mistakes occurring in the development, test, and maintenance processes of these security functional components. Therefore, security cannot in a meaningful manner be expressed as a set of techniques and functions, and assurance aspects of the security products and systems, sometimes referred to as trustworthiness, must be considered. Security assurance results from the performance of appropriate activities or processes to instill confidence that an IT system or product and its security functional components certainly meet their security objectives. Because errors, vulnerabilities and risks will probably always exist over a system's development and life cycle, they will have to be managed within acceptable parameters for specific systems and scenarios. Because threat agents may actively seek to exploit opportunities to violate security polices for illicit gains and to perform disallowed actions on the system, the system should be actively investigated to see whether and to what extent it meets its security objectives and to which level the system is assured and can be trusted. The required security assurance should be achieved to ensure that the developed system is trustworthy for the target scenario.

Consequently, various assurance techniques must be applied in the development of security functional components to provide appropriate evidence that suitable controls have been put in place in the development environment and process. Developers must be able to convince other parties that

- security techniques are selected in a justifiable manner and are suitable to counter foreseeable threats to assets;
- the design and implementation specifications are clear and precise and capture all relevant aspects of the techniques;
- the implementation is a correct and comprehensive instantiation of the design;
- the security functional components are tested for correctness and for attack resistance;
- the development environment is protected from attempts to substitute design or implementation artifacts with malicious components;
- the delivery and installation procedures are such that end users can be assured of receiving an authentic product that can be booted into a secure state;
- the guidance on security features accessible to different users is comprehensive and accurate; and
- the users can trust that the system or product will remain in a secure state if following the guidance.

In brief, security assurance can be derived from two factors: 1) correctness assurance that refers to the assessment of the system or product to verify the correct implementation according to the design; and 2) effectiveness assurance that refers to the suitability of the system's security functions to counter the perceived or identified threats. Assurance techniques and the extent of the rigor to which they are applied produce assurance evidence and thus confidence that the developed system appropriately protects critical assets in the intended environment.

1.3.2 Insufficiencies of Current Grid Security Solutions

Current Grid security solutions have not considered or made arguments about assurance requirements for the trustworthy Grid systems/applications even though some of them may have adopted some common software engineering relevant methods or processes [FOS05]. Consequently, the Grid security functionality cannot be assured even if using theoretically high quality and powerful security techniques and tools in the development. Because systematic evidence was not provided by these Grid security solutions to demonstrate that the development processes were effectively disciplined and the Grid security architectures and designs are appropriate, the implementations may lack enough power of persuasion that they are trustworthy to meet their claimed security objectives and provide claimed strong functionality. Some such implementations may often fail to realize their purposes. Security assurance gives people confidence that the security functionality provided by a Grid system/application is what they really need.

To provide and evaluate assurance for the Grid effectively, it is required that the security functionality is developed in systematic ways [FSH03]. Security assurance requires disciplined engineering, but existing Grid security approaches do not follow any disciplined method or recognized paradigm for specifying security artifacts, implementing security architectures and designing security policies. For a comprehensive, flexible, effective and widely trusted secure Grid environment, the design, development, and implementation of Grid security should be based on a generic security assurance framework, a systematic and disciplined basis for producing appropriate security countermeasures which are highly assured to meet their security objectives for different Grid implementations. In the development following such a Grid security assurance framework, security functional and assurance requirements for the Grid must be derived and analyzed, and consistent design and implementation specifications towards the specific system's realization should also be demonstrated and justified, to provide required assurance evidence.

Even putting security assurance aside, the technical security needs and security functions for the Grid have not been addressed fully and powerfully, either. For example, Globus Toolkit is the most adopted Grid middleware in the world and it describes and sets up a security architecture [KTB05] through the built-in GSI (Grid Security Infrastructure) [FKT98] mechanism for computational Grids; however, there are still some drawbacks and insufficiencies in GSI as well as in other typical Grid security solutions. These need to be overcome and improved for better satisfying various security needs and providing rich security functionality for the evolving Grid. Most current Grid security solutions focus on authentication and pay little attention to authorization. Authentication is to ascertain whether a claimed subject is bona fide while authorization is to decide whether the subject is allowed to perform some actions. Existing Grid security implementations may only use a simple identity based and coarse grained authorization which lacks scalability and flexibility, and the policies to assign privileges to Grid users are vulnerable. Such authorization functions are inadequate for highly expanded and dynamic Grid environments [SSW05] such as the business Grid where many more users, resources, institutions and applications/services are involved and much stronger security functionality and assurance are required. Moreover, there also exist other functional limitations in current Grid security solutions. Few substantial things have been done to address issues such as secure logging, auditing, privacy, and non-repudiation which are often crucial for business and industry applications on the Grid. The systems may require complete and easily understandable log files and effective auditing mechanisms to detect and/or prevent potential or concealed attacks, non-repudiation techniques to resolve disputes, and privacy considerations to protect venders and customers' personal information and impose special constraints on individual resources and services.

To sum up, Grid security solutions in use today only address parts of the security functions, and were not developed in a systematic and disciplined way which is required for producing trustworthy Grid systems/applications, especially commercial and industrial ones. As a result, it is relatively easy to construct Grids that appear secure but fail to preserve their specific security objectives and technical aims.

1.4 Research Objectives

Having considered features and needs of the evolving Grid and recognized the great benefits that Grid computing can bring to people, we believe it is indeed a promising and challenging research field to explore the trustworthy development of highly assured secure Grids by disciplined engineering. The objectives of this project are to address functional and assurance deficiencies of current Grid security solutions, investigate security objectives, requirements and characteristics for the Grid, seek techniques and criteria to design, specify, and justify a generic security assurance framework for the development of trustworthy Grids, and perform a specific high-assurance Grid security implementation following this assurance framework. The implementation will provide technically enhanced security functionality with assurance measures to meet its security objectives. The research aspects are shown below:

- To study and specify disciplined Grid security environment, objectives, functional and assurance requirements and policies at a high level leading to a generic Grid security assurance framework. To date, Grid security researchers have only performed non-disciplined analysis and specification of security artifacts in their security development practices, and designed application-specific security policies and strategies for individual security mechanisms. This approach may lead to complicated and hard to evaluate configurations of Grid security architectures and implementations. Moreover, the components, methods, and/or models used in the implementation can hardly be widely trusted, which prevents their possible reuse. Therefore, a disciplined approach is needed to derive and represent Grid security environment, objectives, and functional and assurance requirements in semi-formal and even formal ways. Some standards and tools can help provide systematic and standardized modeling and specifications, such as ISO/IEC 15408, the Common Criteria (CC) [CC1] [CC2] [CC3], and UML (Unified Modeling Language) [UML].
- To design a justified Grid security assurance framework to include all the disciplined generic security specifications and guide the associated specific design, implementation, test, delivery, operation and maintenance at the specific assurance level, for trustworthy security development. The non-systematic and non-assured implementations of specific security by different current Grid security mechanisms may prevent Grids from being globally adopted by a variety of communities and organizations. The generic Grid security assurance framework needs to effectively cover and address integrated security functional and assurance requirements for the evolving Grid, and the justifications about the suitability and consistency of security specifications, designs, implementations, and tests should also be provided. A development following the generic Grid security assurance

framework will bring a final implementation meeting its security objectives correctly and effectively.

- To improve the Grid security functionality for highly assured implementations. More flexible and powerful authentication and authorization mechanisms are needed to fit new security technical needs of the evolving Grids which are being applied to business and industry applications. The added or improved security functions should work in coordination with other existing Grid security relevant functions in an assured implementation.
- To develop a trustworthy Grid security architecture with implementation following the Grid security assurance framework. The development is a security engineering process. The Grid security implementation will provide improved functionality with required assurance. Because of its disciplined and justified specification and design, the implementation will preserve all dependencies between design artifacts and bring assurance that the claimed security functions are all correctly fulfilled and the security objectives are satisfied. Tests should also be performed to verify the implementation and provide relevant assurance evidence.

1.5 Organization of the Thesis

The rest of this thesis is organized as follows.

Chapter 2 discusses current Grid security solutions and points out their inadequacy for trustworthy Grids. Then several security assurance methods are introduced and compared, and the Common Criteria is finally selected for helping achieve assurance in the security development of Grid systems/applications.

Chapter 3 presents the Grid scenario under research and introduces an exemplar generic Grid-based application execution and information sharing engine as a basis to derive security specifications and construct a Grid security assurance framework, a disciplined approach to developing trustworthy Grid security. An overview of the proposed Grid security assurance framework is then given in this chapter.

Chapter 4 gives the specification of a Common Criteria inspired generic security environment for the Grid, including assets to be protected, subjects of the system, threat agents, assumptions, threats to security, and organizational security policies. The security objectives to counter identified threats and to comply with the assumptions and policies are defined with rationale. Then a set of CC disciplined Grid security functional and assurance requirements are systematically derived and tailored, in order to preserve the security objectives and design the security functions with proper evaluation assurance levels (EALs) [CC3]. Some subsets of the Grid security requirements are also defined to more precisely and concretely fit in with different circumstances, categories and levels of security needs and strengths. All these security specifications are within the generic Grid security assurance framework which guides the development of trustworthy Grids.

Chapter 5 depicts the development from the generic specifications to a specific trustworthy implementation. Following the Grid security assurance framework, a system design summary is specified including the high-level definitions of the security functions and assurance measures for the specific functionality and assurance to be achieved in the development and by the implementation. Then a highly assured implementation of the fine grained security authorization function is described as an example to demonstrate the effective accomplishment of a disciplined security engineering development process.

Associated assurance evidence covering development assurance, life cycle support assurance and tests assurance is also provided.

Chapter 6 concludes the research work that has been done and summarizes future work.

Chapter 2

Grid Security and Security Assurance

2.1 Grid Security

Grid computing not only promotes scientific research such as high-energy physics and life sciences, but also contributes to business, governments and industries. So security functionality and assurance need to be studied to facilitate viable and trustworthy Grid systems/applications, and Grid users must be able to interact in secure, effective and dependable ways. The characteristics that distinguish Grids from traditional distributed systems have led to new security issues to be addressed by the emerging Grid security mechanisms.

2.1.1 Survey of Current Grid Security Solutions

2.1.1.1 GSI of Globus Toolkit 1 and 2

GSI [FKT98] was initially developed within Globus Toolkit 1 and 2 to support distributed computing environments and computational Grids. It supports: creation of temporary proxy credentials; authenticating requests to remote resources; authorization and global-to-local identity mapping; creation of remote processes that may themselves issue further resource requests; and the authentication of inter-process communications [BWE00]. GSI utilizes an X.509 PKI (Public Key Infrastructure) [HPF99] and is layered on top of the

SSLeay/OpenSSL [SSLEAY] [OPENSSL] library which performs the X.509 identity certificate handling and SSL (Secure Sockets Layer) protocol.

GSI focuses primarily and depends heavily on authentication. For the convenience of sharing computational resources across sites, in the authorization GSI provides rather wide privileges to successfully authenticated remote users unless a local system manager has the time to prevent this. Some potential security problems may thus arise. In fact, the simple identity based and coarse grained authorization in GSI may incur new threats to the Grid and make the efforts of a successful authentication void. By this coarse grained authorization an authenticated user can have access to almost any application available from that resource host, which is obviously inappropriate for many Grid environments. Another problem originates from the increase in the number of Grid users along with the number of involved organizations while the number of system administrators at each site remains low. It is difficult for a user to gain access to Grid resources quickly because he has to get many local system managers of these resources to set up local accounts for him, and it is equally difficult to remove this user's access rights from all related Grid resources by the system managers when the rights are no longer valid. Besides, there are also some drawbacks with the user proxy mechanism of GSI which may compromise security: 1) There may exist a private key (for the proxy) on a remote system outside the user's direct control, but it can be used to sign messages which will be trusted by other Grid entities as coming from this user; 2) In creating a proxy certificate, the user is acting like a CA (Certificate Authority) for the proxy, but he does not have a published certification policy or a revocation process. Although Globus seeks to minimize such risks by making proxies short-lived, this represents another way in which the authentication and authorization may be temporarily unreliable in the Grid.

2.1.1.2 GSI of Globus Toolkit 3 and 4

Since OGSA has been specified for the new generation Grid, OGSA Security [SWT02] is also proposed based on the emerging Web Services security models and techniques [WSSEC], and addresses a wider range of security issues than current academic Grids [NJD02]. However, only a few of these issues have been properly implemented. Globus Toolkit 3 [WS03] addresses OGSI, the core components of OGSA, but the new GSI only made security relevant changes in the standard proxy certificates format, in the implemented Web Services protocol, and in the improved resource security model. These modifications are more like a technical improvement than an architectural breakthrough, and most key features of the security architecture of Globus Toolkit 2 are still followed by Globus Toolkit 3. For instance, the mechanisms of user/service identity credentials and coarse grained authorization have not been changed.

As for the recently developed Globus Toolkit 4, its GSI mainly includes a design and implementation update for the existing GT 3 security functions and services, to follow the refactoring and evolution from OGSI to WSRF. A well addressed fine grained authorization mechanism is still unsupported.

2.1.1.3 UNICORE Security Architecture

The UNICORE project [UNICORE] involves large computing centers, public offices, universities, commercial sites and private users. It adopts a PKI implementation different from that of Globus, called U-PKI. The UNICORE security architecture provides for job authentication and secure data transmission by both job signing and data encryption without the use of proxies, and it implements a seamless check of the possible identity certificate chain in the authorization. If there is an invalid certificate within the chain, all subordinated certificates are considered invalid and have to be exchanged or discarded.

However, UNICORE also allows remote users to access the operating system shell and run potential intrusive and perilous code on the resource hosts. Though this can be abated by deploying firewalls, most existing firewalls are statically configured and thus conflict with the dynamic features of Grids. Finally, UNICORE only uses an identity based authorization mechanism which is not flexible and does not scale well.

2.1.1.4 CAS

It is foreseeable that resource owners want to apply different usage strategies for different user groups. Most Grid security solutions, however, are less concerned with authorization and grant authenticated users wide and common privileges. A fine-grained, flexible and scalable authorization mechanism for Grid security is currently lacking.

Efforts to develop suitable authorization mechanisms for the Grid have resulted in the Community Authorization Service (CAS) [PWF02]. The model is built under the assumption that a resource provider will assign controlled access to his resource to a VO, and then the VO can further constrain the ability to VO members individually. A user who wants to access a Grid resource first contacts the CAS server of his VO for a limited X.509 proxy certificate with the rights obtained from the CAS server, and then he uses that certificate to request access to the resource. The resource site checks whether the proxy certificate is genuine and whether the user has the rights required for the access.

CAS provides for the scalability, flexibility, and expressibility of authorization among VOs in the Grid because each resource provider only needs to assign coarse grained rights to a VO. The CAS server of this VO is responsible for the assignment of fine grained privileges to individual VO members. However, there are some problems arising with CAS. First, the user/user proxy can behave as a CA and generate a public/private key pair for the proxy application to use on the user's behalf, which is a problem similar to that of GSI. Secondly, the CAS server is a potential performance bottleneck since no resource can be accessed by the users unless an associated CAS server exists to provide the capabilities, and a successful denial of service attack (e.g. a packet storm) to a CAS server may force all users of that VO to be cut off from the resources. Thirdly, the access control policy is carried in the user's proxy certificate that has to be repeatedly parsed and evaluated by the resource site for each user request. In addition, the approach of limiting the VO's privileges to generate a subset of privileges applicable to an individual VO member presents a subtractive security mechanism and violates the least-privilege principle [SS75].

2.1.1.5 WAS

The Workflow-based Authorization Service (WAS) model was proposed in [KKH03] as an extension of the CAS model with a workflow that is extracted from source code. Although the WAS architecture provides a fine-grained authorization, the workflow may not be available for some processes and it has not been considered in a dynamic VO environment.

2.1.1.6 VOMS

The VOMS (Virtual Organization Management System) [ACC03] [NWC04] model was proposed for authorization in the DataGrid project. In VOMS, group memberships and group rights are managed separately. The VOMS server like CAS manages group memberships which indicate a list of users and roles in each group. Group rights, which are local policies about group memberships, are distributed among resources. They are not stored in the central server. Based upon the roles that have been assigned to the user by the VOMS server, the resource provider decides how much access privilege is granted to the user. So, a user obtains the VOMS server's a priori approval about his group membership and then the resource's approval about group rights.

Similar to the CAS, VOMS focuses on the single VO architecture, so that it has a single point of failure problem and has not considered the dynamic VO environment. Besides, VOMS assumes that group and role information is reported in a format natively understandable to the resource. Therefore, although the system distributes management responsibilities between resource and VO administration, it still presupposes this information is known before the authorization decision may occur.

2.1.1.7 Akenti

Akenti [JMT98] is an authorization system that represents all authorization relevant information for a resource as distributed digitally signed certificates – attribute certificates for the users, and use-condition certificates and policy certificates for the resource. The certificates are independently created by resource stakeholders. When an authorization decision needs to be made, the Akenti system has to gather up all the relevant policy certificates, use-condition certificates and attribute certificates, validate them, and determine the requesting user's privileges to the resource.

Akenti aims at the ability for multiple resource owners and administrative parties to define flexible resource usage policies in a widely distributed system. Nevertheless, it has several obvious shortcomings. First, Akenti does not provide the flexibility in the privilege management mechanism which is required by ad-hoc, short lived collaborations [LK02]. Second, it does not support legacy applications and services. Moreover, since the resource policy is distributed in several policy and use-condition certificates, a resource site or Akenti system must be certain that it has collected all necessary certificates to control access. This is a potential performance bottleneck and hinders the robustness of Akenti. Finally, the authorization relevant certificates are of a proprietary format [TEM03].

2.1.1.8 PERMIS

To overcome the disadvantages of traditional identity based and coarse grained authorization mechanisms, the PERMIS project [PERMIS] [CO02] uses X.509 attribute certificates (ACs) [RFC3281] [X.509] to securely bind and communicate role memberships. It uses policy ACs for the resources and role ACs for the users. Access requests are handled by an access control decision function (ADF) and an access control enforcement function (AEF). ADF and AEF interact through a PERMIS Java API and LDAP (Lightweight Directory Access Protocol) [LDAP] repositories are employed for credential storage.

The PERMIS architecture comprises of a privilege allocation subsystem and a privilege verification subsystem [COB03]. The privilege allocation subsystem is responsible for allocating privileges to the users, and the privilege verification subsystem is for authenticating and authorizing the users. A user accesses resources via an application gateway where the AEF authenticates him and asks the ADF if the user can execute the requested action on the target resource. The ADF has to contact one or more LDAP directories to retrieve the policy AC for the resource and the role AC for the user, and evaluate them to determine whether to grant the request. After receiving a decision from the ADF, the AEF will inform the user of the decision and perform the relevant operations.

Although PERMIS provides flexible authorization, there still exist some potential problems for it: 1) In the PERMIS model, if the centralized LDAP server maintaining policy and role ACs breaks down or is compromised, all resources associated with it will become inaccessible for any user. 2) PERMIS assigns a single SOA (Source of Authority) to manage the creation, update and revocation of policies and roles. Such deployment is fine for small experimental distributed environments with a limited number of users, but when the numbers and hierarchies of users and resources are increasing fast, it will become less efficient to assign all management work to a single SOA. Too centralized control by the SOA will bring many risks.

2.1.2 Security Concerns for Trustworthy Grids

It has been shown in Chapter 1 that security involves both functionality and assurance. However, Grid security solutions introduced in 2.1.1 only consider the functionality and do not follow any disciplined method of security specification, design and implementation. Existing security mechanisms for the Grid provide application-specific development and implementations instead of being based on generic security assurance frameworks. They usually use particular, sometimes even proprietary, security techniques to provide specific security individually. Such ad hoc approaches could achieve potentially good security functionality, but may prevent their wide adoption by Grid communities and impede trusted interactions among different Grid systems and applications. The justification of the implementations and the judgment of whether they meet the security specifications and are suitable to particular use scenarios cannot be assured. A Grid cannot be claimed as trustworthy simply by implementing recognized security techniques such as known cryptographic algorithms and authentication protocols. Most practical attacks may exploit weaknesses in the development and implementations of security measures, not in the properties of the underlying techniques, so that the security cannot be considered appropriate even if the security techniques used are known to be theoretically highly resistant to attacks.

Consequently, effective and efficient security assurance methods need to be followed by Grid security practitioners for achieving widely trustworthy Grids. Security assurance of the Grid refers to the extent of trustworthiness of the Grid security mechanisms, provides evidence of the appropriateness of Grid security specifications and designs, and demonstrates that the Grid security implementations meet the security objectives. The measures used to evaluate assurance concern the ability of the system to reduce the likelihood of vulnerabilities and the extent of the damage that could occur from a vulnerability being exploited.

Besides their neglect of security assurance issues, it is also found that most of the current Grid security solutions are oriented to scientific computation and focus on authentication. Other important security functional features such as authorization, auditing and privacy (to be discussed in section 3.1), only have simple implementations or are even left blank. Such security mechanisms merely satisfy part of the security technical needs for the Grid [NJD02] when it is being taken into business and industry areas. Some exploratory projects have been started trying to enrich the functionality of existing security systems for the Grid, for example, CAS, WAS, VOMS, Akenti, PERMIS and TAS (Ticket-based Fine-grained Authorization Services) [KHK04] to enhance authorization. Nevertheless, the work done still has some deficiencies and is not enough to fulfill many security needs of the evolving Grid scenarios even from the technical view.

In brief, a novel security assurance framework is needed to develop high-assurance Grids with trustworthy and better security functionality.

2.2 Common Criteria and the Grid

Security engineering [AND01] addresses security assurance aspects for the design, implementation, testing, deployment and maintenance of computer systems. An IT security product, system, service, process, or environmental factor (i.e. personnel, organization) or the object of an assurance assessment is called "deliverable" in [ASSUR1]. Since errors, vulnerabilities and risks will probably always exist and may change over the deliverable's life cycle, the task of IT security engineering and management is to manage the security risk by mitigating the vulnerabilities and threats with technological and organizational security measures to achieve a deliverable that performs in the way intended or claimed with acceptable assurance.

2.2.1 Common Criteria – A Security Assurance Method

Assurance considerations must be included for the development of trustworthy Grids as they require disciplined specification, design, and testing towards highly assured security implementations. So security engineering approaches that provide assurance methods should be utilized for the Grid security assurance framework to develop trustworthy Grid security. Because non-standardized security engineering processes (such as [RES04] and [FSH03]) only provide ad hoc assurance methods or incomplete solutions, we explore the Grid security assurance solutions based on international security engineering standards.

An assurance method is a recognized specification for obtaining reproducible assurance results. There exist many IT assurance methods but only a small number of them are specific to IT security. The methods can be categorized into three high-level assurance approaches [ASSUR1]: 1) assessment of the deliverable, i.e., through evaluation and testing; 2) assessment of the processes used to develop or produce the deliverable, where assurance is gained through the inference that the processes implemented affect the quality of the development and implementation of the deliverable and therefore yield corresponding security assurance when applied to the development of IT security deliverables; and 3) assessment of the environment such as the personnel and physical facilities that contribute to the quality of the processes and the production of the deliverable.

Among existing security assurance methods, ISO/IEC standard 15408, known as the Common Criteria [CC1] [CC2] [CC3], is a good option. It is an international standard for specifying and evaluating security properties of IT products and systems and can be used to provide standards and methodology in establishing and specifying a disciplined Grid security assurance framework at a high level. In addition, the Common Criteria is complemented by the Common Evaluation Methodology (CEM) [CEM1] [CEM2]. They provide a harmonized framework and detailed evaluation criteria for IT security evaluation, suitable for both government and general use. From the assurance point of view, the Common Criteria provides an assurance method to assess the deliverable directly over a subset of its life cycle and offers a blend of assurance which is expressed using a range of evaluation assurance levels (EAL, from EAL 1 to EAL 7). The Common Criteria is applicable to both hardware and software products and systems.

Unlike standards such as FIPS 140 [FIPS], Common Criteria does not provide a list of security requirements to be satisfied. Instead, it describes a framework in which users can specify their security requirements, developers can make claims about the security attributes of their products, and evaluators can determine if products actually meet their claims. In other words, Common Criteria provides assurance that the process of specifying, developing, and evaluating an IT security product has been conducted in a rigorous manner.

The Common Criteria has been developed on behalf of a number of governmental information security agencies. It separates consideration of security functionality from security assurance and specifies detailed techniques and functions that can aid in the development of confidence that a security product or system meets its security objectives. The specific assurance techniques and functions are defined in [CC3], and are primarily, but not exclusively, aimed towards assurance obtained through independent assessment or verification. It is intended that consistent application of the evaluation criteria can be verified through national certification schemes.

Within the Common Criteria, functionality and assurance techniques are divided into different areas of applicability, called classes. In each class, different techniques are identified, called families. Each family then identifies one or more levels of rigor by which the technique can be applied; these are called components. For a component, it specifies the precise actions and evidence elements required. A number of packages of assurance components that work together in a complementary manner are defined as EALs. People can construct their TOE (Target of Evaluation, the Common Criteria uses this term, instead of deliverable, to represent an IT product or system and its associated guidance documentation which are the subjects of a security functional and assurance evaluation) security functional and assurance requirements by referring to the content, structure and disciplines of the CC requirements catalogue.

Based on the Common Criteria disciplined implementation-independent security functional and assurance specifications for a category of TOEs, prospective consumers/developers can build or cite a reusable Protection Profile (PP) to express their IT security needs that a compliant product is intended to satisfy, without reference to any specific TOE. A complete Protection Profile should include PP introduction, TOE description, TOE security environment, security objectives, IT security requirements, rationale and possible PP application notes.

For the development of a specific TOE, an implementation-dependent Security Target (ST) for this TOE can be defined conforming to a Protection Profile, to include both its security objectives and requirements and its specific functional summary and assurance measures to meet the stated requirements. The Security Target forms the basis for a possible evaluation.

2.2.2 Other Assurance Methods

Besides the Common Criteria, there are also several other IT standards that provide assurance methods and could be considered for guiding the development of trustworthy security for Grid systems/applications in a disciplined manner. They will be reviewed below.

2.2.2.1 Deliverable Assurance Methods

One recognized standard is FIPS 140-2 [FIPS]. It is used for the validation of cryptographic modules and is commonly used in North America. However, FIPS 140-2 is neither widely recognized elsewhere nor applicable to non-cryptographic modules, which is a fatal limitation for it to be applied for the overall assurance of most computer systems.

The Trusted Computer System Evaluation Criteria (TCSEC) [TCSEC] is a collection of criteria that was previously used to grade or rate the security offered by a computer system product, but there are no new assurance evaluations being conducted using TCSEC at this time. In TCSEC, a class is the specific collection of functionality and assurance requirements to which an evaluated system conforms, and a division is a set of classes. There are four divisions A, B, C, and D and seven classes A1, B3, B2, B1, C2, C1, and D in TCSEC, in decreasing order of features and assurances, and the requirements of a higher class are always a superset of the lower class.

As for Information Technology Security Evaluation Criteria (ITSEC) [ITSEC] and Information Technology Security Evaluation Manual (ITSEM) [ITSEM], they are among the predecessor documents of the Common Criteria and of the Common Evaluation Methodology. They were developed in the early 1990s to provide a framework of evaluation criteria and evaluation methodology for IT security evaluation for the European market. The ITSEC assurance is based on the approach introduced in TCSEC. However, the separation between security functional and assurance requirements in ITSEC allows a greater flexibility. The assurance requirements are themselves again split into the two aspects of effectiveness and correctness. The correctness requirements in ITSEC are presented in the form of six hierarchically ordered assurance levels, the lowest E1 to the highest E6. From a lower level to its adjacent higher level, additional requirements ensure a more rigorous evaluation of the IT product and system. The assurance effectiveness requirements are not included in the assurance levels of ITSEC, but the information obtained from the correctness assessment which is to be used to perform a vulnerability analysis is defined.

2.2.2.2 Process Assurance Methods

A de facto process assurance security engineering standard is SSE-CMM (Systems Security Engineering - Capability Maturity Model) [SSE-CMM] which is the extension of the software engineering standard CMM (Capability Maturity Model) [CMM] in the security area. It aims to improve the organization's systems security engineering processes and to produce a deliverable with capability assurance, and primarily focuses on the process areas (PAs) that are required for systems security engineering, which are referred to as Engineering PAs. The PAs can be tuned to the needs of the organization.

SSE-CMM can be used by organizations to better develop ICT (Information and Communications Technology) security products or deliver ICT security services (such as a threat and risk assessment) with higher quality and within schedule. It is a unique model as it details security engineering requirements for engineering security systems and provides engineering security services in addition to the security requirements, which the development environment must meet. Furthermore, SSE-CMM contains PAs for a wide range of development areas such as defining security requirements, system testing, and threat and vulnerability analysis.

For SSE-CMM, it contains PAs, Base Practices (BP), Generic Practices (GP), and capability levels to describe the organization's processes and to measure how well an organization performs the PAs. The PAs consist of a group of related BPs which focus on specific process activities within an organization while the GPs are related to the overall process maturity throughout the organization. The PAs, BPs and GPs can be thought of as requirements and the capability levels indicate compliance to SSE-CMM; however, these requirements cover what a process achieves but not how the process achieves this end result without interfering with the organizations operating model. A capability level is assigned to each PA to indicate its level of compliance to SSE-CMM, which ranges from Level 0 "Not Performed" to Level 5 "Continuously Improving", and demonstrates that an organization has achieved a minimum capability level for all of the applicable PAs. The GPs are requirements which apply to all PAs related to the overall process maturity and institutionalization of individual PAs throughout the organization. Each successive level indicates an improvement of the organization's overall process maturity and ability to implement the PAs throughout the organization.

Contrasting with the Common Criteria, SSE-CMM focuses on developmental process assurance whereas the CC contains a combination of developmental, analysis, testing, delivery and operation assurance components and thus provides wider scope and more depth for Grid security assurance.

2.2.2.3 Environment Assurance Methods

To achieve high assurance through the environment, the developer's pedigree or personnel assurance is required. The former is the use of prior experience and success rate as an indicator for the quality of security software. Pedigree suggests a method to determine the acceptability of evidence based on the identity of the creator of that evidence using the prior success rate (i.e. track record) that an individual/organization has in developing, maintaining, operating or auditing security products and systems. Personnel assurance ensures that personnel are qualified to fulfill the mission and to mitigate insider threat, and personnel assurance programs are made up of several elements: education, qualification, experience, and integrity of the individual. Among these elements, experience in the IT security field complements qualification and may substitute part of it, and integrity assurance has many contributory elements including: loyalty checks, employment contracts, financial checks, criminal background checks, non-disclosure agreements, etc. Regarding qualification, an exemplar one is CISSP (Certified Information Systems Security Professionals) which is recognized to assure security in operation through certified administrators, operators, and auditors.

ISO 9000 Series provides organizations with an environmental assurance and quality management framework, and allows certification of its successful implementation. It was originally developed for manufacturing organizations, but could be applied to software development organizations by adding relevant interpretation. However, the current version of ISO 9000 Series is still at a sufficiently high level to require further interpretation to be applicable to software development, and it does not address information technology security.

2.2.3 Comparisons

In comparing the three types of assurance methods from the viewpoint of the development of high-assurance secure Grid systems and applications, deliverable assurance methods have some advantages. For process assurance methods, a premise required is that the processes used to design, develop and produce the deliverable are predictable and repeatable. However, because of the highly dynamic, complex and heterogeneous nature of Grids, individual Grid security implementations often have to follow application specific development processes. In contrast, deliverable assurance methods usually focus on the guidance and evaluation of deliverables directly no matter whether they have quite different development processes. Besides, it is relatively easy to investigate deliverables rather than processes, especially when the deliverables were developed based on a security assurance framework. Regarding environment assurance methods, they depend heavily on the recognition and certification of the developers, the personnel expertise and knowledge, and the physical facilities, so that are not always feasible or efficient.

Within the deliverable assurance methods themselves, FIPS 140-2 is not an overall system security engineering standard. As for TCSEC and ITSEC, they are still used in

Europe but are to be phased out by the Common Criteria, which is globally recognized by various agreements [CCRA] [MRA]. Moreover, TCSEC has been expressed as a Common Criteria Protection Profile (PP) [CAPP], and ITSEC assurance levels can be interpreted as Common Criteria EALs [ITSEC-CC-1] [ITSEC-CC-2] [ITSEC-CC-3] [ITSEC-CC-4]. In fact, SSE-CMM and ISO 9000 Series can also be mapped to the Common Criteria [ASSUR3].

Consequently, the Common Criteria is more suitable and effective for the security development of evolving Grids, and can be utilized in the research for constructing a security assurance framework and producing trustworthy Grids.

Chapter 3

Preliminaries

3.1 Research Scope and ClearingHouse Project

This thesis presents the high level specification, analysis and assessment for Grid security and then discusses the relevant implementations carried out in a disciplined way, for the development of trustworthy Grids following a Grid security assurance framework.

Different Grid systems and applications have been investigated, and a typical computational Grid project is the ClearingHouse [CLHOUSE] at Nanyang Technological University, Singapore, collaborating with Osaka University, Japan and Sun Microsystems, Inc. ClearingHouse is an initiative to build a Grid infrastructure to provide a high-level architecture for pervasive and seamless access, especially enterprise-wide access, to heterogeneous computing and information resources/services across multiple administrative domains [CPGRID]. A Grid portal has been developed to provide a set of high level services to enable users to gain access to the resources. Resource management, automatic resource discovery, and economy management of resources in a service-oriented environment are focus issues of the ClearingHouse project currently. An industrial applications project DEGPSE (Design Grid Problem Solving Environments) [DEGPSE] has already been deployed on this infrastructure to provide general analysis and design tools for diverse science and engineering applications. The work is conducted collaboratively by University of Southampton, UK, and Institute of High Performance Computing, Temasek Laboratories and Nanyang Technological University, Singapore.

ClearingHouse addresses the main functions of a typical Grid platform and provides a generic Grid-based application execution and information sharing engine. New issues that emerge with the evolvement of the Grid are also considered in its design. As a result, the ClearingHouse project provides us with a basis and reference for exploring disciplined security environment, objectives and requirements for the computational Grid infrastructure and portal, to construct a generic security assurance framework for the development of trustworthy Grid security. Below, several common security technical needs for a generic Grid-based application execution and information sharing engine and its environment are described informally, and can be referred to for drawing the disciplined Grid security objectives and functional requirements and designing the security functions. These security needs are presented in an order similar to that of the Common Criteria functional classes which will be introduced in the next section.

- Security audit: Recognizing, recording, storing and analyzing the information about security relevant activities should be performed to help secure the Grid systems/applications.
- Non-repudiation: The identity of any party involved in a data exchange should be assured. This means that the originator cannot deny having sent the message, nor can the recipient deny having received it.
- User data protection and privacy: Sensitive Grid user relevant information should be protected as confidential and having integrity.

- Authentication: It should be verified that the user is genuinely the one he claims to be when he requests access. Also, resources affiliated to the system need to be verified as trusted secure ones.
- Authorization: The user must be assigned appropriate privileges to utilize the specific resources of the system before he can perform any action on them.
- Security management: The important information about the system, application, and resources should be properly handled and can only be operated on by authorized people.
- Protection, interoperability, and compatibility of the security mechanism: The security component of a system must be itself secure enough to be able to deploy the protection for other system components effectively. The security implementation has to be robust and well designed in widely compatible and easily interoperable manners on secure hardware and software.
- Secure communications: Trusted channels/paths should be available to deliver data between or within the environment and the system.

3.2 Methodology and Overview of the Proposed Grid Security Assurance Framework

A proposed generic security assurance framework is shown in Figure 3.1 for the development of trustworthy Grids. It is disciplined and validated according to the Common Criteria, and the disciplined Grid security specifications consist of a security environment, security objectives, IT security functional and assurance requirements as well as relevant rationale which are systematically derived for a computational Grid TOE such as the ClearingHouse and Nanyang Campus Grid [CPGRID]. Then conforming to the generic Grid security specifications, different TOE summaries can be specified for different specific TOE implementations to produce individual highly assured Grid security deployments. These are instantiations of the security specifications and TOE summaries making use of specific security and Grid techniques and tools with specific assurance measures, to suit different use scenarios and security needs following the Grid security assurance framework. The security objectives rationale, security requirements rationale, and TOE summary specification rationale provided within the Grid security assurance framework and disciplined development processes will demonstrate the conformance and justify the appropriateness of the security specifications and high-level definitions. This security assurance framework is a hierarchical one of security specification, design, implementation, test, deployment and operation that delivers an approach for developing trustworthy Grids.

3.2.1 Stage 1 – Establish Security Environment

In this framework, the TOE security environment is a narrative statement of the security problems to be solved by the Grid TOE and must therefore be defined first. It describes the security aspects of the environment in which the Grid TOE and the TOE assets are intended to be used and the manner in which the TOE is expected to be employed. The TOE security environment addresses threats to security, organizational security policies, and usage assumptions. The detailed process of establishing a security environment for the Grid is demonstrated in section 4.1.

3.2.2 Stage 2 – Identify Security Objectives

Security objectives are concerned with the Grid TOE and its environment, and reflect the stated intent to counter identified threats and comply with organizational security policies and assumptions. The security objectives rationale demonstrates that the objectives address all of the environmental aspects identified and are effective and provide a suitable coverage. Section 4.2 describes how we derive and justify security objectives for the Grid.

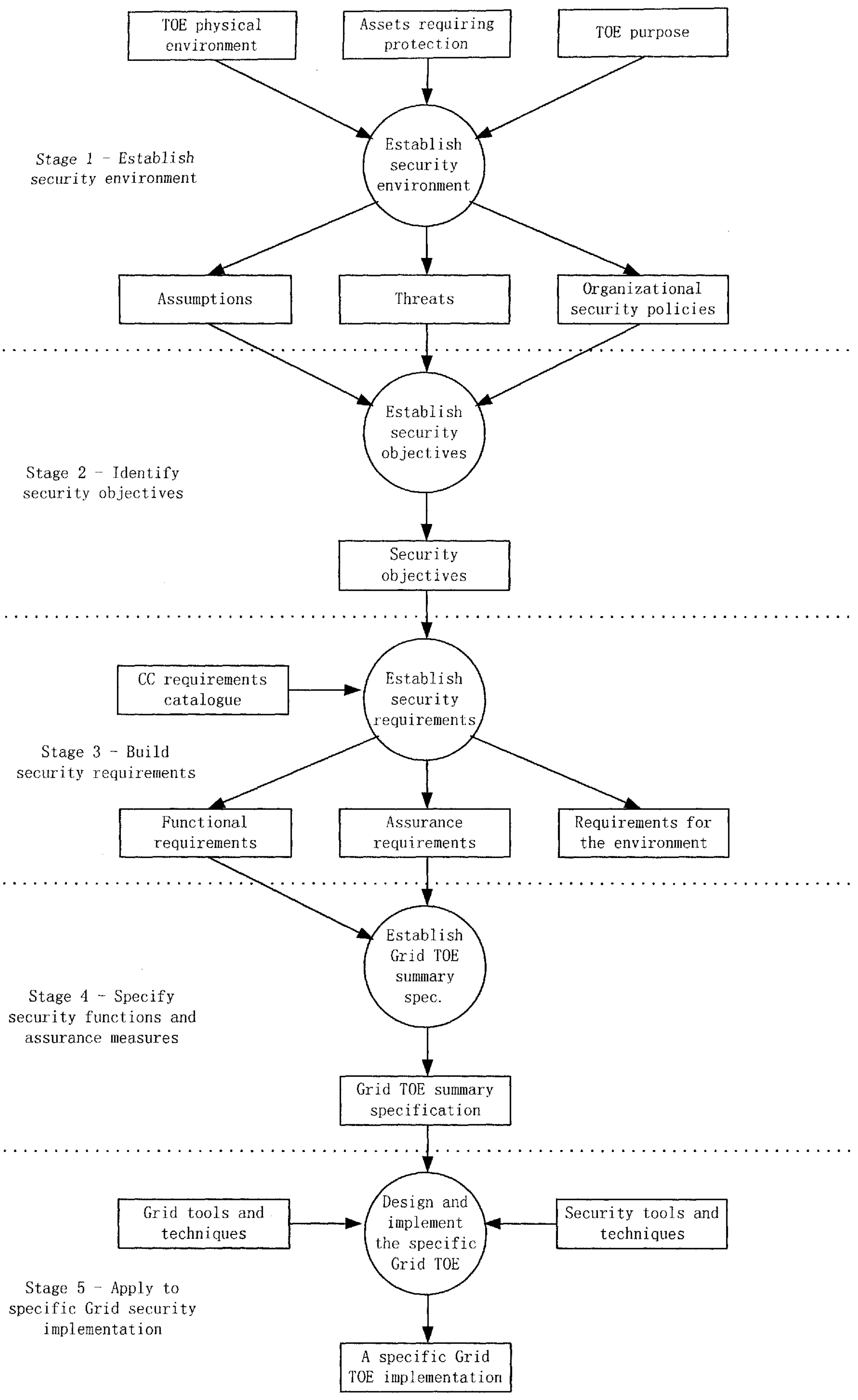
TOE physical environment
Assets requiring protection
TOE purpose
Establish security environment
Stage 1 - Establish security environment
Assumptions
Threats
Organizational security policies
Establish security objectives
Stage 2 - Identify security objectives
Security objectives
CC requirements catalogue
Establish security requirements
Stage 3 - Build security requirements
Functional requirements
Assurance requirements
Requirements for the environment
Establish Grid TOE summary spec.
Stage 4 - Specify security functions and assurance measures
Grid TOE summary specification
Grid tools and techniques
Design and implement the specific Grid TOE
Security tools and techniques
Stage 5 - Apply to specific Grid security implementation
A specific Grid TOE implementation

Figure 3.1 A Security Assurance Framework for the Development of Trustworthy Grids

3.2.3 Stage 3 – Build Security Requirements

Following the security environment and security objectives, specified IT security requirements for the Grid TOE will include both security functional requirements and security assurance requirements. Security functional requirements are levied on those functions of the TOE that are specifically in support of IT security and define the desired security behaviors of the TOE, and security assurance requirements may specify a minimum strength level consistent with the security objectives claimed. An optional statement can be included to identify the security requirements for the environment of the Grid TOE. The security requirements rationale demonstrates that the set of disciplined security requirements are suitable to meet and are traceable to the security objectives.

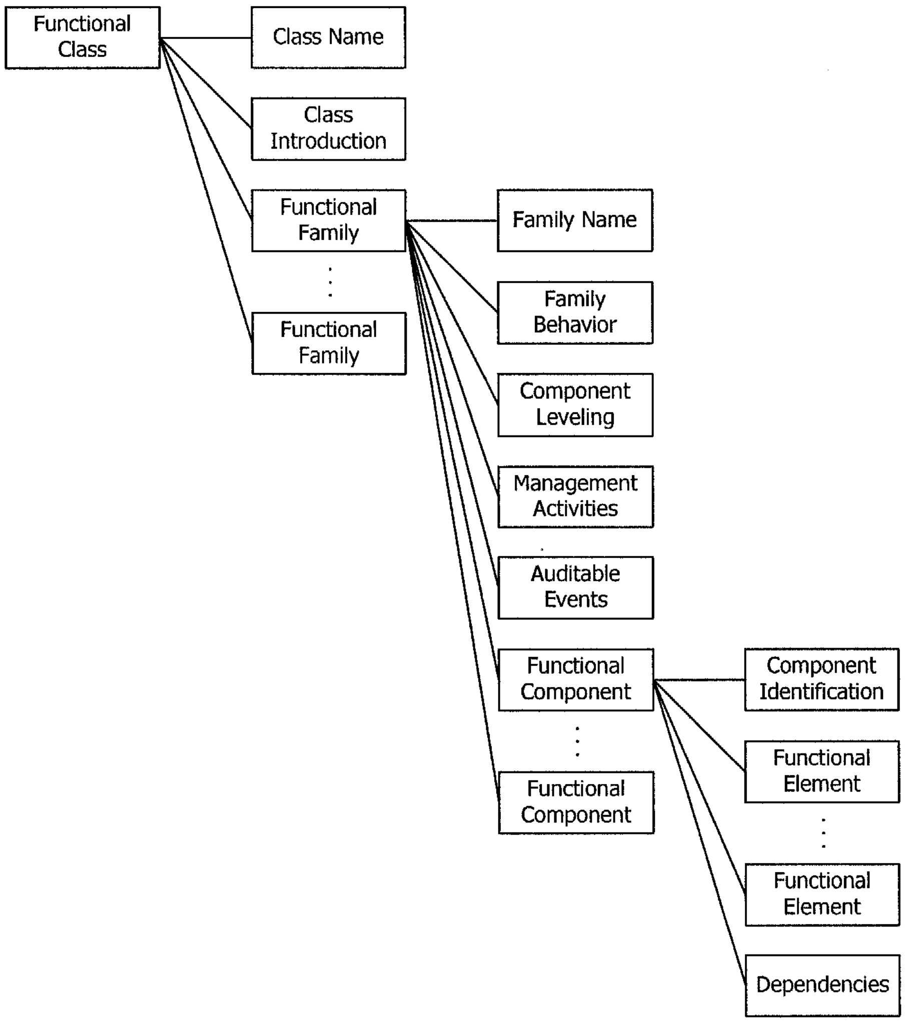

Figure 3.2 The Structure of the Expressions of Grid Security Functional Requirements

For the CC compliant security functional requirements for the Grid TOE and for its IT environment, they can be distributed among seven functional classes: 1) Class FAU (Security Audit); 2) Class FDP (User Data Protection); 3) Class FIA (Identification and Authentication); 4) Class FMT (Security Management); 5) Class FPR (Privacy); 6) Class FPT (Protection of the TSF (TOE Security Functions)); and 7) Class FTA (TOE Access). Here the term class is used for the most general grouping of security requirements. All members of a class (e.g. Class XYZ) share a common focus, while differing in coverage of security objectives. A class specification can consist of class name, class introduction and several functional families. As for a family (e.g. Family XYZ_ABC), it is a grouping of sets of security requirements that share security objectives but may differ in emphasis or rigor. Some functional components plus family name, family behavior, component leveling, management requirements and audit requirements constitute a functional family specification. Here a functional component (e.g. Component XYZ_ABC.1) describes a specific set of security requirements and is the smallest selectable set of security requirements for inclusion in the structures defined according to the Common Criteria. The set of functional components within a family may be ordered to represent increasing strength or capability of security requirements that share a common purpose. They may also be partially ordered to represent related non-hierarchical sets. Finally, the components are constructed from individual functional elements. The indivisible element (e.g. Element XYZ_ABC.1.1) is the lowest level expression of security requirements and the smallest security requirement identified and recognized by the Common Criteria. Each functional component specification may include component identification, dependencies among the components, and at least one functional element. Dependencies among functional components will arise when the functional component is not self sufficient and has to rely upon another functional component's proper functioning. Figure 3.2 depicts the structure and terminology of all levels of disciplined expressions of the security functional requirements. The CC compliant Grid security assurance requirements (such as those on configuration management, product delivery and operation, development design and implementation, guidance document, life cycle support, tests and vulnerability assessment) will be expressed in a similar way - as assurance classes, families, components and elements.

With the basic CC disciplined security requirements for the Grid TOE, some further operations may even be made on the security functional requirements in order for addressing different aspects of the security needs of the TOE in more detail. The security requirements can be tailored by using permitted operations to meet a specific security objective, enforce a specific security policy, or counter a specific threat. The operations are selected from the following set:

- Iteration: It allows a component to be used more than once with varying operations for expressing multiple security needs, policies, and so on. If necessary, to cover different aspects of the same requirement (e.g. identification of more than one type of user), repetitive use of the same component to cover each aspect is permitted.
- Assignment: It allows the specification of parameters of a security functional requirement to be filled in when the functional component is used. Some components have elements that contain parameters that enable the developer to specify a set of values for incorporation into the specification to meet a security objective, and these elements clearly identify both each parameter and the constraint on values that may be assigned to that parameter. Any aspect of an element whose acceptable values can be unambiguously described or enumerated can be represented using a parameter. The parameter may be an attribute or rule that narrows the requirement to a specific value or range of values. For instance,

based on a security objective, an element within a functional component may state that a given action should be performed for a number of times. In this case, the assignment would provide the number, or range of numbers, to be used for the parameter.

- Selection: It allows the specification of one or more items from a list in a functional element. This is the operation of picking one or more items from a list in order to narrow the scope of an element within a functional component.
- Refinement: It allows the addition of extra details to the security requirements when they are used. For all functional components, it is permitted to limit the set of acceptable implementations by specifying additional details in order to meet some specific security objectives. For example, a few technical details can be added if necessary.

For Grid security assurance requirements, only the iteration and refinement operations are applicable to be performed. Section 4.3 includes the details of all the CC disciplined Grid security functional and assurance requirements and associated rationale.

3.2.4 Stage 4 – Define Specific Security Functions and Assurance Measures – TOE Summary Specification

The security environment, security objectives, and security functional and assurance requirements may even be applicable for a Grid scenario or class instead of a single Grid TOE if they follow systematic and disciplined derivation, specification and justification. For a specific security development of high-assurance Grid, a TOE summary specification of the target product is further needed. It provides a high-level definition of the target TOE's security functions claimed to meet the functional requirements and the assurance measures taken to meet the assurance requirements. The TOE summary specification defines the instantiation of the CC disciplined security requirements for a specific Grid TOE under development, and is applied to a trustworthy security implementation. An exemplar TOE summary specification for a specific Grid TOE is defined in section 5.1.

3.2.5 Stage 5 – Apply to Specific Grid Security Implementations

The accomplishment of a specific trustworthy Grid TOE and the practical justification of its compliance with the proposed Grid security assurance framework are demonstrated in the TOE's final design and implementation. The TOE should be realized based on the security functional requirements and the TOE's summary specification. The TOE implementation is accomplished using a process of applying security engineering and Grid skills and knowledge and enforcing a variety of assurance measures, and the resulting TOE will meet its specific security objectives if it correctly and effectively fulfills all of its security functional and assurance requirements. In the research, we not only need disciplined security specifications and rationale for the development of trustworthy Grid TOEs, but also practical implementations of highly assured products to complete the security engineering processes following the Grid security assurance framework. Consequently, relevant practices, experiments, tests and documentation should be carried out for the implementation of the trustworthy Grid TOEs to fulfill the security specifications and designs and verify the fulfillment. Section 5.2 provides an overview of a specific highly assured security implementation for the Grid TOE following the TOE summary specification defined in section 5.1, and sections 5.3 to 5.5 provide detailed assurance evidence for that development and implementation.

In the proposed Grid security assurance framework, there are mainly three parties involved in the development: TOE consumers, TOE developers, and TOE evaluators.

Consumers specify the security needs and even define a PP (Protection Profile), developers follow these to build a ST (Security Target) and accordingly perform the practical security design and implementation, and evaluators will finally check the security functions and assurance measures to judge the conformance of the implemented TOE to its security requirements. To ensure that an engineering process is compatible with the proposed CC compliant security assurance framework, we can verify the development and product following the CEM (Common Evaluation Methodology) [CEM1] [CEM2] and relevant national CC schemes such as CCEVS (The NIAP Common Criteria Evaluation and Validation Scheme for IT Security) [CCEVS] and Singaporean Common Criteria Scheme (being established).

3.3 Protection Profile, Security Target and the Disciplined Security Specifications for the Grid

As shown in subsection 2.2.1, the CC disciplined security environment, objectives, and requirements can be used as standardized implementation-independent security specifications to build a Protection Profile (PP) for a category of TOEs. To extend this work into a complete Grid Protection Profile, in addition to the disciplined generic Grid security specifications discussed in section 3.2, a PP introduction including PP identification and PP overview plus a TOE description are needed. The security objectives rationale and security requirements rationale contained in the PP may also require more detailed coverage, sufficiency, suitability and consistency analysis of the security specifications, justification of the dependencies and mutual support among security requirements, and other relevant information. The rationale supports the claims that a conformant Grid TOE will provide an effective set of IT security countermeasures within the security environment. Finally, some PP application notes should also be included in the Grid PP if necessary.

Concerning a Security Target produced conforming to the Grid PP for a future specific TOE development, it needs to include an ST introduction which consists of ST identification, ST overview and CC conformance, a TOE description, and the PP claims besides the TOE security environment, objectives, requirements and summary specification. PP claims contain the claim that the TOE conforms to the requirements of one or more PPs, and present an explanation, justification and supporting material which may include reference to the PP, a PP refinement statement and a PP additions statement. The ST's rationale should involve detailed justifications of the coverage, sufficiency, suitability and consistency for the specific target Grid TOE's summary specification and the PP claims in addition to those for the security objectives and requirements. The rationale demonstrates that the ST is a complete and cohesive set of requirements, that a conformant TOE would provide an effective and suitable set of countermeasures within the security environment, that the TOE summary specification addresses the requirements, and that the PP conformance claims are valid.

Chapter 4

Disciplined Generic Security Specifications for High-Assurance Grids

4.1 Security Environment for the Grid

Here the security environment is established for a computational Grid TOE such as the ClearingHouse, as the starting point for trustworthy security development and for deriving security objectives and further security artifacts following the security assurance framework. The security environment is the context in which the TOE is intended to be used, and the threats to security that are, or are held to be, present in the environment are also included. For a security environment, all aspects of the TOE's physical environment which are relevant to TOE security should be identified, including subjects and known physical and personnel security arrangements. The purpose of a TOE must be clarified to address its type and intended usage in the environment, and the TOE assets requiring protection by the security component of the TOE have to be specified. The protections are security functions and assurance measures on which the security requirements and policies are imposed. At the same time, assumptions should be stated. They cannot be enforced by technical means but have to be met by the environment in order for the TOE to be considered secure. Some organizational security policies may also be needed to identify relevant rules and policies. In brief, the security environment for the Grid will mainly address: 1) assets; 2) subjects; 3) usage assumptions; 4) organizational security policies; and 5) threats to security. They are presented in a disciplined manner in the following subsections.

4.1.1 Assets

AST.ResourceData	Information about the resources/services, including resource/service descriptions, statuses, policies, etc
AST.UserData	User relevant information such as identity and role/attribute credentials and resource usage quota
AST.JobData	Job data including code, status and results
AST.InfoResource	Information resources/services of the Grid TOE used to share knowledge
AST.CompResource	Computing resources/services of the Grid TOE
AST.AuditData	Audit data such as log files recording security relevant user and system activities
AST.SecurityMechanism	Security and administrative mechanisms of the TOE

Table 4.1 The Data Dictionary of Assets

Assets are listed in Table 4.1 where AST.ResourceData can include the resource/service name, owner/institution, location of the resource/service host, registration time, and other resource/service information such as the total and current computing power of the computing resource/service, the directory and index of the information resource/service, and the availability and policy of the resource/service. For AST.UserData, the genuine user identity can be represented using the identifier and technique adopted by existing authentication systems such as X.509 PKI. AST.UserData includes user accounts which record users' quotas to bid for resources and should be protected as confidential and having integrity. Intermediate and final results of a job are both AST.JobData. AST.InfoResource can be a biological gene bank, meteorological database, and/or application code deposit which should be protected from unauthorized disclosure, modification, and deletion. As regards AST.CompResource, they are used to run user programs and need to be protected from illegal use which is herein confined to use by unauthorized users, but the protection can be extended to prevent use by authorized users for unauthorized purposes. Finally, it should be noted that AST.AuditData can only be accessed by authorized subjects, and AST.SecurityMechanism themselves must be protected as robust and secure.

4.1.2 Subjects

S.User	End user of the TOE
S.Admin	S.User who manages and controls the TOE
S.ResourceProvider	S.User who contributes his own resource to the TOE
S.CommonUser	S.User other than S.Admin and S.ResourceProvider
S.Attacker	Threats agent to the TOE

Table 4.2 The Data Dictionary of Subjects

A subject is an entity within the set of interactions occurring with or within a TOE that causes operations to be performed. The set of interactions is subject to a set of rules that regulate how assets are managed, protected and distributed within the Grid TOE. The S.User listed in Table 4.2 can be identified as an S.Admin, an S.ResourceProvider, or an S.CommonUser. S.Admin performs TOE administrative functions for both resources and users of the Grid TOE, and he can also make use of all other TOE functions. S.ResourceProvider only performs appropriate administrative functions for his own resource in the TOE such as managing his resource data without conflicting with TOE administration, and he can use other TOE functions authorized for him. For S.CommonUser, he can only use some non-administrative TOE functions under proper control of the TOE security mechanism. S.Attacker shown in Table 4.2 is a human, or a process acting on his behalf, located outside the TOE. S.Attacker wants to access sensitive data and resources in the Grid TOE illegally and if he succeeds, he may steal, tamper, misuse, or even destroy them.

4.1.3 Usage Assumptions

There is one usage assumption for the security environment of the TOE, as shown in Table 4.3:

A.SecureKey	The device or application to generate and store security keys/tokens and other relevant data which are used in the secure communication, authentication and authorization for the users and the Grid TOE, is protected when outside the control of the TOE and is of high quality.

Table 4.3 The Data Dictionary of Assumptions

4.1.4 Organizational Security Policies

Organizational security policies demonstrated in Table 4.4 should be enforced for the TOE to preserve secure state:

P.ResourceLocal Security	For a resource wanting to join or already existing in the Grid TOE, the resource's local security guards provided by the resource host, other than the protection provided by the TOE, effectively prevent the possibility that the resource is compromised at the host level by the attacker without being found.
P.StandardCredentials	Widely adopted and easily compatible standards, formats and parameters are used to represent users' identity/role/attribute credentials and encode other security relevant information, for this information to be used and evaluated effectively and efficiently.
P.HardwareSecurity	The hardware of the TOE is protected in order to resist physical attack.

Table 4.4 The Data Dictionary of Organizational Security Policies

4.1.5 Threats to Security

Among the threats listed in Table 4.5, T.Hack_Phys may result in arbitrary security compromises, so this threat can endanger all the assets. For T.ResourceData_Unsecured, it is possible that an attacker modifies, disables, or even deletes some resource data of information resources/services in order to deny their use by authorized users of the TOE, or to enable them to be used by illegal subjects. Similarly, an attacker may modify resource data of computing resources/services to prevent them from being correctly searched and scheduled by authorized users or the TOE. Computing resources/services can thus be disabled from executing authorized user jobs, and illegal user jobs may be permitted to run on them. The realization of T.UserData_Forgery could lead an attacker to be successfully granted access to the information resources/services and be allowed to execute his jobs on the computing resources/services of the Grid TOE. T.JobData_Unsecured may prevent jobs from being correctly and effectively executed, checked and handled. Finally, by the fulfillment of T.AuditData_Unsecured, an attacker can adjust his attack activity according to the disclosed audit data of the TOE to escape possible security check, and he may even modify or delete some log files to conceal his attacks.

T.Hack_Phys	An attacker interacts with the TOE interfaces to exploit all possible vulnerabilities in the physical environment.
T.ResourceData_ Unsecured	An attacker extracts and releases sensitive resource data of the TOE without authorization. He may modify or even delete some resource data.
T.UserData_ Unsecured	An attacker extracts and releases sensitive TOE user relevant information illegally. He may even modify and/or delete AST.UserData without appropriate privileges.
T.UserData_Forgery	An attacker forges valid user identity, role/attributes, usage quota, and/or other possible security relevant AST.UserData.
T.JobData_ Unsecured	An attacker extracts and releases sensitive user job data of the TOE without necessary privileges, and he may modify or even delete job data.
T.UserData_JobData _Cheat	An attacker forges a TOE interface to defraud the user of his sensitive AST.UserData and AST.JobData.
T.InfoResource_ IllegalUse	An attacker illegally reads, modifies, and/or deletes the contents of the information resources/services which are intended to be shared among authorized users of the TOE only.
T.CompResource_ IllegalUse	An attacker makes use of the TOE to run his programs on the computing resources/services without right authorization.
T.InfoResource_ CompResource_ Insecure	An attacker impersonates a trusted resource provider to the TOE and contributes insecure, potentially intrusive information/computing resource/service to the Grid.
T.AuditData_ Unsecured	An attacker steals audit relevant information of the TOE and even modifies or deletes some AST.AuditData.

Table 4.5 The Data Dictionary of Threats to Security

4.2 Security Objectives for the Grid

Based on the security environment for the TOE, a clear statement of the security objectives to eliminate those identified threats to security needs to be made. Consequently, this section identifies and defines security objectives for the Grid TOE and its environment. The security objectives will reflect the stated intent and counter the identified threats, and they should comply with the organizational security policies and usage assumptions as well as the operational aim, product purpose and physical environment of the system. These security objectives need also to be followed by the detailed specifications of the CC disciplined security functional requirements and assurance requirements which will be grouped for designing security functions and assurance measures for the Grid. The security objectives provide a systematic basis for devising security countermeasures for different Grid implementations.

4.2.1 Security Objectives for the TOE

Table 4.6 shows a collection of basic security objectives for the computational Grid TOE like the ClearingHouse. Among these objectives, OT.AccessControl consists of two aspects. One aspect is AC_Enforced: access control routines including authentication, authorization and other possible functions should be involved to check each relevant access to the TOE, and the other aspect is AC_NonBypassible: successfully passing the evaluation performed by the access control routines is the only way for outside users to get access to the TOE resources, and attempts to bypass the access control routines will not lead to any successful access to the resources. For OT.UserData_Secured, an S.CommonUser can only view his own usage quota about the resources. In OT.InfoResource_CompResource_LegalUse, the TOE should provide functions to detect or prevent any unauthorized read, modification, or deletion attempt to its information resources/services, and provide functions to detect any attacking attempt to its computing resources/services as well as to prevent those attacks. The attacks can include unauthorized use of the computing resources/services to run programs that may impede, disable, and even destroy normal operation and sharing of the resources.

OT.Auth_TOE	The TOE should enable the user to identify it as the true TOE he wants to interact with.
OT.AccessControl	The TOE can only be used by properly authenticated and authorized users.
OT.ResourceData_ Secured	Resource data of the TOE should be protected to be read and operated on by appropriately authorized TOE users only.
OT.UserData_Secured	Sensitive user relevant data used to verify his genuine identity and validate his privileges can be assured as confidential and having integrity. Also AST.UserData involved in resource trading should be managed and adjusted only by the system or S.Admin.
OT.JobData_Secured	Job relevant data should be protected and can only be retrieved, modified, and deleted by appropriately authorized users of the TOE.
OT.InfoResource_ CompResource_ LegalUse	The contents of the information resource/service of the TOE can only be read, modified, and deleted by authorized users. The computing resource/service can only be used to run jobs by authorized users.
OT.InfoResource_ CompResource_Trusted	The Grid TOE only contains trusted, secure and available information and computing resources/services.
OT.AuditData_Secured	Audit relevant data of the TOE is confidential and tamper-resistant. Only properly authorized users can read and utilize AST.AuditData.
OT.TSM_Secure	The security and administrative mechanisms of the TOE should be protected securely to work effectively.
OT.Lifecycle_Security	The TOE shall detect flaws during the initialization, personalization and operational usage.

Table 4.6 The Data Dictionary of Security Objectives for the TOE

4.2.2 Security Objectives for the Environment of the TOE

There exist four derived security objectives to be addressed by the environment of the Grid TOE. They are listed in Table 4.7:

OE.SecureComm	The communications between users and the TOE and within the TOE should be secure – intact or even confidential.
OE.UserData_ LocalProtection	It requires the user side to keep secure the held AST.UserData which will be used to request and get access to the TOE.
OE.StandardFormat	It requires that the representation formats of security relevant data such as credentials for the users and the TOE should be commonly available and compatible.
OE.Underlying Protection	The underlying hardware and software for the Grid TOE should be protected as secure.

Table 4.7 The Data Dictionary of Security Objectives for the Environment of the TOE

4.2.3 Security Objectives Rationale

Threats-Assumptions-Policies/ Security Objectives	OT.Auth_TOE	OT.AccessControl	OT.ResourceData_Secured	OT.UserData_Secured	OT.JobData_Secured	OT.InfoResource_CompResource_LegalUse	OT.InfoResource_CompResource_Trusted	OT.AuditData_Secured	OT.TSM_Secure	OT.Lifecycle_Security	OE.SecureComm	OE.UserData_LocalProtection	OE.StandardFormat	OE.UnderlyingProtection
T.Hack_Phys		X							X	X			X	X
T.Resource Data_ Unsecured		X	X								X			
T.UserData_ Unsecured		X		X							X	X		
T.UserData_ Forgery		X		X						X		X		
T.JobData_ Unsecured		X			X						X			
T.UserData_ JobData_ Cheat	X													
T.Info Resource_ IllegalUse		X				X								
T.Comp Resource_ IllegalUse		X				X								
T.Info Resource_ Comp Resource_ Insecure							X		X	X		X		X
T.AuditData_ Unsecured		X						X	X	X				
A.SecureKey											X	X		
P.Resource Local Security							X		X	X				X
P.Standard Credentials		X											X	
P.Hardware Security		X							X					X

Table 4.8 Security Environment to Security Objectives Mapping for Generating Tracings

A rationale is needed to demonstrate the appropriateness and correspondence of the security specifications. As a result, Table 4.8 is presented to express the tracings between the security environment artifacts and the security objectives for a generic Grid-based application execution and information sharing engine. The stated security objectives should be traceable to all the aspects identified in the TOE security environment and are suitable to cover them. A more detailed suitability analysis about the objectives for the policies, threats, and assumptions also needs to be performed. Following this rationale, people can explore more effectively and efficiently from security environment to security objectives, and potentially to security requirements and functions.

4.3 Security Requirements for the Grid

After specifying the security environment and security objectives, a set of security functional and assurance requirements for the Grid TOE and its IT environment should be defined and justified subject to the Common Criteria. These describe the desired security behavior by the TOE and its IT environment, in order to preserve the security objectives and allow the design of the security functions with proper EALs. Security requirements are a refinement of the security objectives and if satisfied, will ensure that the TOE can fulfill these objectives with the required EAL. The disciplined Grid security requirements describe security properties preserved by the Grid TOE and its IT environment, and users can detect these properties by direct interaction with the IT or by the IT response to stimulus. The satisfaction of the security functional requirements will provide the claimed security features, and security assurance requirements impose needs for the appropriateness and correctness of these security features. In brief, the security requirements provide a systematic basis of specific security solutions for implementing suitable and highly assured security for a generic Grid-based application execution and information sharing engine in its design, implementation, testing and evolution.

4.3.1 Security Functional Requirements for the Grid

From the security objectives derived in section 4.2 and the informally summarized security technical needs for the Grid which were mentioned in Chapter 3, a set of Grid security functional requirements are systematically drawn whose specifications are disciplined conforming to the Common Criteria Part 2 [CC2].

4.3.1.1 Specification of Grid Security Functional Requirements

The CC disciplined security functional requirements for the generic Grid-based application execution and information sharing engine are described in this part. Basic information of all families and components of each functional class is presented in Table 4.9, but more detailed description of the functional elements and other relevant information are not specified because of the limited thesis length.

- **Security Audit (Class FAU) -** The security audit class addresses security functional requirements of the Grid TOE that involve recognizing, recording and storing information related to security relevant user and system activities. The resulting audit records can be examined to determine when and which security relevant activities took place and who is responsible for them.
- **User Data Protection (Class FDP) -** This class contains families specifying requirements for TOE security functions and TOE security function policies related to protecting user access related data. The protection should address relevant user, job, resource and other possible information within the TOE, during import and storage, and security attributes that are directly related to the data.
- **Identification and Authentication (Class FIA) -** This functional class addresses the requirements for TOE security functions to establish and verify a claimed user identity. Identification and authentication are to ensure that the users are associated with proper security attributes (user data), e.g. identity, groups, roles,

security or integrity levels. Other classes of Grid security functional requirements, such as Class FDP and Class FAU, are dependent upon correct identification and authentication of the users in order to be effective.

- **Security Management (Class FMT)** - Class FMT is intended to specify the management of several aspects of the TSF: security attributes, TSF data and functions. The different management roles and their interactions, such as separation of capability, can also be specified. The scope of the fulfillment of this class includes the management of TSF data, management of security attributes (e.g. resource access control policies, user capability lists), management of functions in TSF, and definition of security roles. The TSF data form the administrative databases that guide and affect the enforcement of the TSP (TOE Security Policy). Examples of the TSF data include audit information and system and TSF configuration parameters.
- **Privacy (Class FPR)** - This class contains privacy requirements that provide user protection against the discovery and misuse of his identity by other people.
- **Protection of the TSF (Class FPT)** - The functional class FPT contains both the functional requirements that relate to the integrity and management of the mechanisms that provide the TSF (independent of TSP-specifics), and the functional requirements that relate to the integrity of TSF data (independent of the specific contents of the TSP data).
- **TOE Access (Class FTA)** - This functional class specifies security functional requirements for controlling the establishment of a user's session.

Security Functional Family	Security Functional Components and Elements
FAU_ARP Security audit automatic response	FAU_ARP.1 Security alarms – .1
FAU_GEN Security audit data generation	FAU_GEN.1 Audit data generation – .1, .2
	FAU_GEN.2 User identity association - .1
FAU_SAR Security audit review	FAU_SAR.1 Audit review – .1, .2
	FAU_SAR.2 Restricted audit review - .1
	FAU_SAR.3 Selectable audit review - .1
FAU_SEL Security audit event selection	FAU_SEL.1 Selective audit - .1
FAU_STG Security audit event storage	FAU_STG.1 Protected audit trail storage – .1, .2
FDP_ACC Access control policy	FDP_ACC.1 Subset access control - .1
FDP_ACF Access control functions	FDP_ACF.1 Security attribute based access control – .1, .2, .3, .4
FDP_DAU Data authentication	FDP_DAU.2 Data authentication with identity of guarantor – .1, .2
FDP_ITC Import from outside TSF control	FDP_ITC.2 Import of user data with security attributes – .1, .2, .3, .4, .5
FDP_RIP Residual information protection	FDP_RIP.1 Subset residual information protection - .1
FDP_SDI Stored data integrity	FDP_SDI.2 Stored data integrity monitoring and action – .1, .2
FIA_AFL Authentication failures	FIA_AFL.1 Authentication failure handling – .1, .2
FIA_ATD User attribute definition	FIA_ATD.1 User attribute definition - .1
FIA_UAU User authentication	FIA_UAU.1 Timing of authentication – .1, .2
	FIA_UAU.3 Unforgeable authentication – .1, .2
	FIA_UAU.5 Multiple authentication mechanisms – .1, .2
	FIA_UAU.6 Re-authenticating - .1
FIA_UID User identification	FIA_UID.1 Timing of identification – .1, .2
FIA_USB User-subject binding	FIA_USB.1 User-subject binding - .1
FMT_MOF Management of functions in TSF	FMT_MOF.1 Management of security functions behavior - .1
FMT_MSA Management of security attributes	FMT_MSA.1 Management of security attributes - .1
	FMT_MSA.2 Secure security attributes - .1
FMT_MTD Management of TSF data	FMT_MTD.1 Management of TSF data - .1
	FMT_MTD.3 Secure TSF data - .1
FMT_REV Revocation	FMT_REV.1 Revocation – .1, .2
FMT_SAE Security attribute expiration	FMT_SAE.1 Time-limited authorization – .1, .2
FMT_SMF Specification of management functions	FMT_SMF.1 Specification of management functions - .1
FMT_SMR Security management roles	FMT_SMR.2 Restrictions on security roles – .1, .2, .3
FPR_ANO Anonymity	FRP_ANO.1 Anonymity - .1
FPT_AMT Underlying abstract machine test	FPT_AMT.1 Abstract machine testing – .1
FPT_FLS Fail secure	FPT_FLS.1 Failure with preservation of secure state - .1
FPT_RPL Replay detection	FPT_RPL.1 Replay detection – .1, .2
FPT_RVM Reference mediation	FPT_RVM.1 Non-bypassability of the TSP - .1
FPT_STM Time stamps	FPT_STM.1 Reliable time stamps - .1
FPT_TDC Inter-TSF TSF data consistency	FPT_TDC.1 Inter-TSF basic TSF data consistency – .1, .2
FTA_LSA Limitation on scope of selectable attributes	FTA_LSA.1 Limitation of scope of selectable attributes - .1
FTA_MCS Limitation on multiple concurrent sessions	FTA_MCS.2 Per user attribute limitation on multiple concurrent sessions – .1, .2
FTA_TAB TOE access banners	FTA_TAB.1 Default TOE access banners - .1
FTA_TAH TOE access history	FTA_TAH.1 TOE access history – .1, .2, .3
FTA_TSE TOE session establishment	FTA_TSE.1 TOE session establishment - .1

Table 4.9 Security Functional Requirements for the Grid

4.3.1.2 Tailoring Security Functional Requirements

Some useful tailoring operations have also been performed on the disciplined Grid security functional requirements which were presented in 4.3.1.1, to meet security objectives and counter threats more specifically, but because of the limited length of this thesis, only two examples are given below.

One example is about the functional component FDP_ACC.1 *Subset access control* of the family FDP_ACC. FDP_ACC identifies the access control relevant SFPs (Security Functional Policies) and defines the scope of control of the policies that form the identified access control portion of the TSP. The rules that define the functionality of a specific access control SFP will be set in the specification of other functional families such as FDP_ACF *Access control functions* and FDP_SDI *Stored data integrity.* FDP_ACC for the Grid TOE only includes the component FDP_ACC.1 *Subset access control* because FDP_ACC.2 *Complete access control*, according to its definition from the Common Criteria, is very hard to implement and prove in practice. No management activities or auditable events are foreseen for this family. As for FDP_ACC.1, it has the dependency on FDP_ACF.1 *Security attribute based access control.* Since multiple access control policies may be defined and can be accomplished by iterating relevant functional components for each named policy, the specification for FDP_ACC.1 is iterated five times here for developing trustworthy Grid security: enforcing Authentication SFP, Authorization SFP, Management SFP, TrustworthyResource SFP, and SecureCommunication SFP respectively. Different assignments are also imposed on individual iterations:

- Access control policy (FDP_ACC)
    - FDP_ACC.1 Subset access control

- FDP_ACC.1.1 (original): The TSF shall enforce the [assignment: access control SFP] on [assignment: list of subjects, objects, and operations among subjects and objects covered by the SFP]
- FDP_ACC.1.1/Authentication SFP (after iteration and assignment): The TSF shall enforce the [Authentication SFP] on [access requests submitted by S.User to the TOE]
- FDP_ACC.1.1/Authorization SFP (after iteration and assignment): The TSF shall enforce the [Authorization SFP] on [S.User's requests to run jobs on the computing resources/services, to operate on the information resources/services, or to handle AST.UserData or AST.JobData in the TOE]
- FDP_ACC.1.1/Management SFP (after iteration and assignment): The TSF shall enforce the [Management SFP] on [security management relevant operations on resource data, audit data and TOE security mechanisms. These operations can only be performed by S.Admin or S.ResourceProvider]
- FDP_ACC.1.1/TrustworthyResource SFP (after iteration and assignment): The TSF shall enforce the [TrustworthyResource SFP] on [whether to allow a resource contributed by S.User to join the Grid, and if the resource is admitted, its provider will become S.ResourceProvider]
- FDP_ACC.1.1/SecureCommunication SFP (after iteration and assignment): The TSF shall enforce the [SecureCommunication SFP] on [parameters involved in the user session establishment and interface for TOE access, and data exchanged between the TOE and the outside and within the TOE]

The other example is on FTA_TAH.1 *TOE access history*. It is the only component of the functional family FTA_TAH which defines requirements for the TSF to display to a user, upon successful session establishment, a history of successful and unsuccessful attempts to access the TOE using the user's name. There are no management activities or auditable events foreseen for FTA_TAH. For FTA_TAH.1, it has no dependency on other functional components, and has three functional elements. Selection operations have been performed on FTA_TAH.1.1 and FTA_TAH.1.2:

- TOE access history (FTA_TAH)
  - FTA_TAH.1 TOE access history
    - FTA_TAH.1.1 (original): Upon successful session establishment, the TSF shall display the [selection: date, time, method, location] of the last successful session establishment to the user
    - FTA_TAH.1.1 (after selection of all items): Upon successful session establishment, the TSF shall display the [date, time, method, location] of the last successful session establishment to the user
    - FTA_TAH.1.2 (original): Upon successful session establishment, the TSF shall display the [selection: date, time, method, location] of the last unsuccessful attempt to session establishment and the number of unsuccessful attempts since the last successful session establishment
    - FTA_TAH.1.2 (after selection of all items): Upon successful session establishment, the TSF shall display the [date, time, method, location] of the last unsuccessful attempt to session establishment and the number of unsuccessful attempts since the last successful session establishment

- FTA_TAH.1.3 (original): The TSF shall not erase the access history information from the user interface without giving the user an opportunity to review the information

4.3.1.3 Security Functional Requirements for the IT Environment

Apart from the security functional requirements for the Grid TOE, some other security requirements can also be specified to be met by the IT environment of the TOE if required, such as those for non-repudiation in the communication, cryptographic support, physical protection, resource utilization and trusted paths/channels.

4.3.1.4 Common Security Technical Needs and Disciplined Security Functional Requirements for the Grid

Common Grid Security Technical Needs/Disciplined Grid Security Functional Requirements	Class FAU	Class FDP	Class FIA	Class FMT	Class FPR	Class FPT	Class FTA	Environmental Functional Requirements
Security audit	X			X				X
Non-repudiation								X
User data protection and privacy		X		X	X			X
Authentication		X	X	X		X		X
Authorization		X	X	X		X		
Security management	X	X	X	X		X		
Protection, interoperability, and compatibility of the security mechanism				X		X	X	X
Secure communications								X

Table 4.10 Common Technical Needs to Disciplined Requirements Correspondence

Table 4.10 shown above provides a correspondence between the common Grid security technical needs informally summarized in Chapter 3 and the CC disciplined security functional requirements for the Grid TOE and its IT environment. It can be seen that all basic security technical needs for the generic Grid-based application execution and information sharing engine have been addressed properly by the disciplined security specifications in the Grid security assurance framework.

4.3.2 Security Assurance Requirements for the Grid

Security assurance requirements for the Grid TOE are expressed in similar structures as security functional requirements - assurance classes, families, components and elements, and possible dependencies among assurance components should also be addressed. Usually assurance components coming from different assurance classes and families are grouped into certain EALs that balance the level of assurance obtained with the cost and feasibility of acquiring that degree of assurance. There are seven hierarchically ordered EALs [CC3] defined in the Common Criteria for the rating of a TOE's assurance. Each EAL includes no more than one component of each assurance family because all components in an assurance family are hierarchically related and a higher assurance component covers all issues of a lower assurance component. The increase in assurance from EAL 1 to EAL 7 is accomplished by the substitution of a hierarchically higher assurance component from the same assurance family (i.e. increasing rigor, scope, and/or depth) and/or the addition of assurance components from other assurance families (i.e. adding new requirements).

Among these EALs, EAL 1 - *functionality tested* provides low assurance. Regarding EAL 5 – *semiformally designed and tested*, EAL 6 – *semiformally verified design and tested*, and EAL 7 - *formally verified design and tested*, they involve many semiformal and formal design, specification, test and justification relevant issues which are difficult to realize currently. Consequently, EAL 1, 5, 6, and 7 may not be very appropriate to be imposed for current and emerging Grids. It is considered that EAL 2 - *structurally tested*, EAL 3 - *methodically tested and checked*, and EAL 4 - *methodically designed, tested, and* *reviewed*, can be adopted as a basis to guide implementing and evaluating assurance in the security development of a generic Grid-based application execution and information sharing engine. It is obvious that the security assurance requirements, expressed as an EAL, must be flexible depending on the application scenarios, so different EALs may be chosen and employed for different specific Grid TOE implementations on which the users or developers require different assurance and trustworthiness. The three EALs include standard assurance components disciplined by the Common Criteria, and can be augmented by adding new assurance components from higher EALs.

1. Evaluation Assurance Level 2 – Structurally Tested

EAL 2 requires the developer to deliver some design information and test results, but should not demand more effort on the part of the developer than is consistent with good commercial practice. As such it should not require a substantially increased investment of cost or time in the development. Therefore, EAL 2 is applicable in those circumstances where developers or users require a low to moderate level of independently assured security in the absence of ready availability of the complete development record. Such a situation may arise when securing legacy systems, or where access to the developer may be restricted.

At EAL 2, it provides assurance by an analysis of the security functions, using a functional and interface specification, guidance documentation and the high level design of the Grid TOE, to understand the security behavior. The analysis is supported by independent testing of the TOE security functions, evidence of developer testing based on the functional specification, selective independent confirmation of the developer test results, strength of function analysis, and evidence of a developer search for obvious vulnerabilities (e.g. those in the public domain).

The assurance components included in this EAL are demonstrated in Table 4.11.

Assurance Class	Assurance Components of EAL 2
Configuration Management	ACM_CAP.2 Configuration items
Delivery and Operation	ADO_DEL.1 Delivery procedures
	ADO_IGS.1 Installation, generation, and start-up procedures
Development	ADV_FSP.1 Informal functional specification
	ADV_HLD.1 Descriptive high-level design
	ADV_RCR.1 Informal correspondence demonstration
Guidance Documents	AGD_ADM.1 Administrator guidance
	AGD_USR.1 User guidance
Tests	ATE_COV.1 Evidence of coverage
	ATE_FUN.1 Functional testing
	ATE_IND.2 Independent testing – sample
Vulnerability Assessment	AVA_SOF.1 Strength of TOE security function evaluation
	AVA_VLA.1 Developer vulnerability analysis

Table 4.11 Assurance Components included in EAL 2

2. Evaluation Assurance Level 3 – Methodically Tested and Checked

EAL 3 permits a conscientious developer to gain maximum assurance from positive security engineering at the design stage without substantial alteration of existing sound IT development practices. It is applicable in those circumstances where developers or users require a moderate level of independently assured security, and requires a thorough investigation of the TOE and its development without substantial re-engineering.

This EAL provides assurance also by an analysis of the security functions, using a functional and interface specification, guidance documentation, and the high level design of the TOE, but the analysis is supported by independent testing of the TOE security functions, evidence of developer testing based on both the functional specification and the high-level design, selective independent confirmation of the developer test results, strength of function analysis, and evidence of a developer search for obvious vulnerabilities.

EAL 3 represents a meaningful increase in assurance from EAL 2 by requiring more complete testing coverage of the security functions and mechanisms and/or procedures that provide some confidence that the TOE will not be tampered with during development. Table 4.12 contains assurance components of EAL 3.

Assurance Class	Assurance Components of EAL 3
Configuration Management	ACM_CAP.3 Authorization controls
	ACM_SCP.1 TOE CM coverage
Delivery and Operation	ADO_DEL.1 Delivery procedures
	ADO_IGS.1 Installation, generation, and start-up procedures
Development	ADV_FSP.1 Informal functional specification
	ADV_HLD.2 Security enforcing high-level design
	ADV_RCR.1 Informal correspondence demonstration
Guidance Documents	AGD_ADM.1 Administrator guidance
	AGD_USR.1 User guidance
Life Cycle Support	ALC_DVS.1 Identification of security measures
Tests	ATE_COV.2 Analysis of coverage
	ATE_DPT.1 Testing: high-level design
	ATE_FUN.1 Functional testing
	ATE_IND.2 Independent testing – sample
Vulnerability Assessment	AVA_MSU.1 Examination of guidance
	AVA_SOF.1 Strength of TOE security function evaluation
	AVA_VLA.1 Developer vulnerability analysis

Table 4.12 Assurance Components included in EAL 3

3. Evaluation Assurance Level 4 – Methodically Designed, Tested, and Reviewed

EAL 4 permits a developer to gain maximum assurance from positive security engineering based on good commercial development practices which, though rigorous, do not require substantial specialist knowledge, skills and other resources. It is therefore applicable in those circumstances where developers or users require a moderate to high level of independently assured security in conventional commodity TOEs and are prepared to incur additional security-specific engineering costs.

At this assurance level, assurance will be provided by an analysis of the security functions, using a functional and complete interface specification, guidance documentation, the high-level and low-level design of the Grid TOE, and a subset of the implementation, to understand the security behavior. Assurance is additionally gained through an informal model of the TOE security policy. The analysis should be supported by independent testing of the TOE security functions, evidence of developer testing based on the functional specification and high-level design, selective independent confirmation of the developer test results, strength of function analysis, evidence of a developer search for vulnerabilities, and also an independent vulnerability analysis demonstrating resistance to penetration attackers with a low attack potential.

Assurance Class	Assurance Components of EAL 4
Configuration Management	ACM_AUT.1 Partial CM automation
	ACM_CAP.4 Generation support and acceptance procedures
	ACM_SCP.2 Problem tracking CM coverage
Delivery and Operation	ADO_DEL.2 Detection of modification
	ADO_IGS.1 Installation, generation, and start-up procedures
Development	ADV_FSP.2 Fully defined external interfaces
	ADV_HLD.2 Security enforcing high-level design
	ADV_IMP.1 Subset of the implementation of the TSF
	ADV_LLD.1 Descriptive low-level design
	ADV_RCR.1 Informal correspondence demonstration
	ADV_SPM.1 Informal TOE security policy model
Guidance Documents	AGD_ADM.1 Administrator guidance
	AGD_USR.1 User guidance
Life Cycle Support	ALC_DVS.1 Identification of security measures
	ALC_LCD.1 Developer defined life-cycle model
	ALC_TAT.1 Well-defined development tools
Tests	ATE_COV.2 Analysis of coverage
	ATE_DPT.1 Testing: high-level design
	ATE_FUN.1 Functional testing
	ATE_IND.2 Independent testing – sample
Vulnerability Assessment	AVA_MSU.2 Validation of analysis
	AVA_SOF.1 Strength of TOE security function evaluation
	AVA_VLA.2 Independent vulnerability analysis

Table 4.13 Assurance Components included in EAL 4

In brief, EAL 4 represents a meaningful increase in assurance from EAL 3 by requiring more design description, a subset of the implementation, and improved mechanisms and/or procedures that provide confidence that the TOE will not be tampered with during development or delivery. The assurance components contained in this EAL are presented in Table 4.13.

Detailed contents and specifications of the seven basic EALs can be found in [CC3].

4.3.3 Security Requirements Rationale

In order to state the rationale demonstrating the appropriateness and correspondence of the security requirements and objectives and help justify the development following the Grid security assurance framework, Table 4.14, a mapping table of Grid security functional requirements and security objectives, is used to express intuitively, clearly, and unambiguously the tracings between them and to show that the requirements are suitable to meet the objectives. This mapping table can even be extended to include the security functional requirements and security objectives for the IT environment of the Grid TOE. A detailed sufficiency and suitability analysis has also been made for the functional requirements. For example, FMT_MTD *Management of TSF data* is involved for the fulfillment of OT.AuditData_Secured, OT.AccessControl and OT.TSM_Secure, and FPT_RVM *Reference mediation* is needed for OT.AccessControl to ensure the non-bypassability of the access control routines and TSP enforcement functions. Tracings can be further given from the tailored security functional components and elements to the security objectives for demonstrating the suitability and coverage of the disciplined Grid security requirements in more detail. Two such examples, one about the tailored FDP_ACC.1 *Subset access control* and the other about the tailored FTA_TAH.1 *TOE access history*, are shown in Table 4.15.

TOE Security Functional Requirements/ TOE Security Objectives	OT.Auth_TOE	OT.AccessControl	OT.ResourceData_Secured	OT.UserData_Secured	OT.JobData_Secured	OT.InfoResource_CompResource_LegalUse	OT.InfoResource_CompResource_Trusted	OT.AuditData_Secured	OT.TSM_Secure	OT.Lifecycle_Security
FAU_ARP		X								
FAU_GEN		X								X
FAU_SAR		X						X		X
FAU_SEL		X							X	
FAU_STG								X		X
FDP_ACC	X	X	X	X	X	X	X	X	X	
FDP_ACF	X	X	X	X	X	X	X	X	X	
FDP_DAU							X		X	X
FDP_ITC		X							X	
FDP_RIP			X							X
FDP_SDI			X	X	X					
FIA_AFL		X								
FIA_ATD		X							X	
FIA_UAU		X								
FIA_UID		X								
FIA_USB		X								
FMT_MOF		X							X	
FMT_MSA		X	X	X	X				X	
FMT_MTD		X						X	X	
FMT_REV		X							X	X
FMT_SAE		X					X		X	X
FMT_SMF									X	
FMT_SMR		X							X	
FPR_ANO		X		X					X	
FPT_AMT									X	X
FPT_FLS									X	X
FPT_RPL		X								
FPT_RVM		X								
FPT_STM		X							X	
FPT_TDC									X	X
FTA_LSA		X								
FTA_MCS		X								
FTA_TAB		X								
FTA_TAH		X								
FTA_TSE		X								

Table 4.14 Security Objectives to Security Functional Requirements Mapping

Tailored TOE Security Functional Requirements/TOE Security Objectives	OT.Auth_TOE	OT.AccessControl	OT.ResourceData_Secured	OT.UserData_Secured	OT.JobData_Secured	OT.InfoResource_CompResource_LegalUse	OT.InfoResource_CompResource_Trusted	OT.AuditData_Secured	OT.TSM_Secure	OT.Lifecycle_Security
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
FDP_ACC.1.1/ Authentication SFP	X	X								
FDP_ACC.1.1/ Authorization SFP		X		X	X	X				
FDP_ACC.1.1./ Management SFP		X	X					X	X	
FDP_ACC.1.1/ TrustworthyResource SFP							X			
FDP_ACC.1.1/Secure Communication SFP				X	X					
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
FTA_TAH.1.1 (after selection operation)		X								
FTA_TAH.1.2 (after selection operation)		X								
FTA_TAH.1.3 (after selection operation)		X								
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 4.15 Security Objectives to Tailored Security Functional Requirements Mapping

Concerning the security assurance requirements, it has been discussed in subsection 4.3.2 why EAL 2, 3, and 4 are selected for flexible development of trustworthy Grid security implementations. The fulfillment of the security assurance requirements and EALs will cover all the security objectives.

In addition, it needs to be checked whether the security functional and assurance requirements dependencies for the Grid TOE are completely fulfilled. For any security requirement on which another Grid security requirement depends, the former should already be a functional or assurance requirement included within the set of the disciplined Grid security requirements to be satisfied by the Grid TOE. FDP_ACC.1 *Subset access control* can be used as an example again here, as it has a dependency on FDP_ACF.1 *Security attribute based access control.* For FDP_ACF.1, it is also a functional component included for the security requirements of the Grid TOE. Because all dependencies of each security functional/assurance component for the Grid TOE are presented along with the detailed specification of the component, the full analysis of the completeness, appropriateness, and satisfaction of the dependencies among Grid security requirements can be easily performed but is not included here. In the mean time, a dependency analysis helps demonstrate the mutual support and internal consistency of the security functional requirements and the security assurance requirements.

For the specific TOEs developed following the generic Grid security assurance framework and disciplined security specifications, different security functional strengths may even be required on different TOEs. For instance, the enforcement of Authentication SFP, Authorization SFP, and Management SFP is needed for the fundamental protection of assets of a Grid TOE, so FDP_ACC.1/Authentication SFP, FDP_ACC.1/Authorization SFP, and FDP_ACC.1/Management SFP should be addressed for basic, medium, and high level security. For circumstances where only attackers possessing a low attack potential exist, most possible breaches of security are casual rather than straightforward, intentional or deliberately planned. In such cases, identification and authentication by regular means will provide appropriate confidence in a resource owner to contribute a secure resource to the Grid, and it is enough to only provide integrity protection, without encrypting all transmitted data, for the information exchanged between the TOE and the outside subjects and within the TOE. Consequently, FDP_ACC.1/TrustworthyResource SFP and FDP_ACC.1/SecureCommunication SFP are only needed for medium and high level security of the Grid TOE.

For the example of FTA_TAH.1, it is used to prompt the user with his previous access related information and help him find possible access attempts by opponents imitating him or having stolen/forged his user information. This function could be ignored in the security mechanisms of the Grid TOE which are applied to the scenarios where there are no attackers with high attack potentials or deliberately organized attempts to breach security, so FTA_TAH.1 is only assigned on the high level security functionality for the Grid TOE implementations.

Therefore, different subsets of the Grid security functional requirements and their tailored specifications could be defined for providing more flexible security deployments, sometimes even with different EALs. As discussed before, EAL 2, 3, and 4 are currently suitable for the development of a trustworthy generic Grid-based application execution and information sharing engine. They can be applied respectively for the above described security circumstances and associated basic, medium, and high level security functionalities, to provide their corresponding assurance. A full grouping of the tailored Grid security functional requirements for different security strengths and EALs of the TOE implementations is shown in Table 4.16, but considering the thesis length, its full derivation processes are not detailed here except the two examples concerning FDP_ACC.1 and FTA_TAH.1 discussed in the above two paragraphs.

Security Functional Requirements	Basic	Medium	High
Class FAU			
FAU_ARP	1	1	1
FAU_GEN (tailored)	1,2	1,2	1,2
FAU_SAR/S.Admin	1,2	1,2	1,2,3
FAU_SAR/S.ResourceProvider	1,2	1,2	1,2,3
FAU_SEL (tailored)	NA	NA	1
FAU_STG (tailored)	1	1	1
Class FDP			
FDP_ACC/Authentication SFP	1	1	1
FDP_ACC/Authorization SFP	1	1	1
FDP_ACC/Management SFP	1	1	1
FDP_ACC/TrustworthyResource SFP	NA	1	1
FDP_ACC/SecureCommunication SFP	NA	1	1
FDP_ACF/Authentication SFP	1	1	1
FDP_ACF/Authorization SFP	1	1	1
FDP_ACF/Management SFP	1	1	1
FDP_ACF/TrustworthyResource SFP	NA	1	1
FDP_ACF/SecureCommunication SFP	NA	1	1
FDP_DAU (tailored)	NA	1	2
FDP_ITC (tailored)	NA	2	2
FDP_RIP	NA	NA	1
FDP_SDI	2	2	2
Class FIA			
FIA_AFL (tailored)	NA	1	1
FIA_ATD	1	1	1
FIA_UAU (tailored)	1,3	1,3	1,3,5,6
FIA_UID	1	1	1
FIA_USB	1	1	1
Class FMT			
FMT_MOF	1	1	1
FMT_MSA/Authorization SFP	1	1	1,2
FMT_MSA/Management SFP	1	1	1,2
FMT_MTD (tailored)	1	1	1,3
FMT_REV (tailored)	1	1	1
FMT_SAE	1	1	1
FMT_SMF	1	1	1
FMT_SMR (tailored)	1	2	2
Class FPR			
FPR_ANO	NA	NA	1
Class FPT			
FPT_AMT	NA	NA	1
FPT_FLS	NA	1	1
FPT_RPL	NA	NA	1
FPT_RVM	1	1	1
FPT_STM	NA	NA	1
FPT_TDC	NA	NA	1
Class FTA			
FTA_LSA (tailored)	NA	1	1
FTA_MCS	NA	1	2
FTA_TAB	NA	NA	1
FTA_TAH (tailored)	NA	NA	1
FTA_TSE	1	1	1
Evaluation Assurance Level	2	3	4

Table 4.16 Grouping Security Functional Requirements for Different Security Strengths

Following the security requirements rationale, people can explore more effectively from Grid security objectives to security functional and assurance requirements and show that the requirements are suitable to meet the security objectives. The rationale can also be utilized in guiding the design of security functions and assurance measures for further TOE summary specifications and trustworthy implementations.

4.4 Summary

In this chapter, disciplined specifications of the generic security environment, objectives, and functional and assurance requirements are derived and justified for the Grid following a security engineering approach, a CC validated security assurance framework for trustworthy Grid development. The generic Grid security functional requirements can, if implemented to meet the security assurance requirements dictated by the EAL, properly preserve the security objectives of the Grid, and can even be further refined, extended, and grouped for providing different security strengths and achieving different EALs which are needed by different scenarios or specific implementations.

Highly assured specific security architectures and implementations for the Grid can thus be designed and realized systematically and flexibly. They will be derived in a manner that preserves all dependencies between design artifacts and brings assurance that all security relevant TOE and environmental elements are considered. The specification, tailoring, and grouping of the security functional requirements and the functional classes can lead to diverse designs of the security functions for specific TOE summaries and implementations, along with the specific assurance measures designed following the security assurance requirements. In a TOE summary specification for a specific Grid security implementation following the Grid security assurance framework, an instantiation of the security requirements should be performed for the specific Grid TOE and a high-level definition of the security functions and assurance measures that meet the functional and assurance requirements should be given. The assurance will be obtained from the evidence and confidence in the correctness of the design and implementation and the effectiveness of the security functions and assurance measures. A sample TOE summary specification and highly assured implementation for a specific Grid TOE is demonstrated in the next chapter.

Chapter 5

From Generic Specifications to Specific Implementation

Disciplined generic security specifications have been derived within the CC validated security assurance framework for the development of trustworthy Grids. To further demonstrate the appropriateness of the specifications and apply them in a practical development following the security assurance framework, a subsequent implementation-dependent TOE summary specification and its associated highly assured security design and implementation for the Grid TOE are required. The implementation will also bring practical experience and knowledge for better understanding security requirements and recursively refining the specifications and framework.

5.1 A Grid TOE Summary Specification

To implement a highly assured specific Grid TOE and security functionality following the generic Grid security specifications and assurance framework, a high level definition and refinement is first needed for all specific security functions and assurance measures of the target TOE based on the generic Grid security functional and assurance requirements, and needs to be included in this specific TOE's summary specification. The summary will be justified and can be subsequently used to conduct the lower level design and implementation of a specific security architecture for the generic Grid-based application execution and information sharing engine. Thus the Grid security functionality will be realized with high assurance in the systematic way that we have adopted for the disciplined development.

5.1.1 Security Functions Summary

1. Security Audit (SF.Audit)

This security function is to recognize, record, and store information related to security relevant activities. The resulting audit data can be analyzed to determine what activities took place and which user is responsible for them, and possible responsive actions can be further performed by the system or S.Admin. For instance, when a potential security violation is detected, the security audit function is able to take alarm actions, such as notifying the administrator, and/or to close the session. Proper audit information should be recorded in an easily understandable format and stored in a protected way. Only appropriately authorized users are able to view and/or operate on the audit data of the TOE through the security audit function.

2. Authentication and Secure Communication (SF.Authen_Comm)

This security function is to verify a claimed user identity as well as to provide underlying secure communication support. The effective functioning of other security functions is dependent upon correct authentication.

A powerful and widely accepted authentication mechanism, such as the X.509 PKI, should be adopted by this security function to represent, deliver, and validate user identity and cryptography relevant information. Secure communication can thus be provided. The function will also monitor the number and time of unsuccessful authentication attempts by a user, and if a threshold is reached, the user session will be terminated and the user account be locked or the point of entry from which the attempts were made be blocked. This authentication and communication security function should be able to detect the use of authentication relevant data which has been forged or copied.

3. Authorization (SF.Authorization)

An authenticated user should be evaluated by this security function to determine his privileges on accessing specific resources and/or relevant data of the Grid TOE. The evaluation is based on the user request, user identity, user role/attributes and the resource's access policy. If authorized, the remote user should be appropriately and effectively bound to a local subject acting on behalf of him and be provided with the proper handle to the resource/data.

4. Access Control and Interface (SF.AC_Interface)

SF.AC_Interface governs the interface and access to the Grid TOE and its resources. It should be non-bypassable. This security function addresses the invocation and enforcement of SF.Authen_Comm and SF.Authorization, and only successfully authenticated and authorized users can make use of TOE resources. Moreover, possible replay attack using identified entities (e.g. messages, requests, user information, and sessions) will be detected and subsequent actions will be performed to counter the attack. The number of concurrent user sessions and the scope of the session security attributes can also be limited by this function, and if required for privacy, the real user name of a requesting S.User can even be made invisible to other S.Users (excluding S.ResourceProvider of the requested resource and S.Admin).

5. Security Management and Data Protection (SF.Mgmt_DP)

This security function involves the protection and management of AST.UserData, AST.ResourceData, AST.JobData, and the TOE security mechanism (AST.SecurityMechanism) itself.

The AST.UserData (e.g. authentication and authorization relevant user information and user's resource usage quota), AST.ResourceData (e.g. resource access policies and resource status information), AST.JobData (e.g. program code and execution results) and relevant system data should be stored and managed securely without unauthorized access and operation. Also the previous content of them may need to be made unavailable in order to fulfill FDP_RIP *Residual information protection*. The security management and data protection function can only be controlled by the administrators of the TOE or the resource, and could be enabled to preserve a secure state in the face of identified failures. This function will work in coordination with other security relevant functions of the Grid TOE and ensure the required consistency (e.g. data's consistency, reliable time's consistency) among them.

In addition, consideration for the dependability of the Grid resources should be included in the TOE security management. The information and computing resources/services participating in the Grid TOE must be secure and dependable, therefore SF.Mgmt_DP needs to check each resource when it applies to join the Grid and monitor the resource when it is operating within the Grid. If an existing resource in the Grid is no longer trusted, it should be expelled. The guarantee that only dependable resources are running within the Grid is based on and supported by the TOE's organizational security policies and the resources' local security guards.

5.1.2 Assurance Measures Summary

To summarize Grid TOE assurance measures that achieve EALs lower than EAL 5, people can simply use a general mapping from the assurance documentation/evidence which will be produced in the disciplined development process, to the appropriate corresponding Grid security assurance requirements.

1. Configuration Management Assurance (Class ACM)

For basic level security, EAL 2 is imposed and it includes ACM_CAP.2 *Configuration items*. Regarding medium level security and EAL 3, ACM_CAP.3 *Authorization controls* and ACM_SCP.1 *TOE CM coverage* should be satisfied. If high level security is required, EAL 4 will be imposed including ACM_AUT.1 *Partial CM automation*, ACM_CAP.4 *Generation support and acceptance procedures*, and ACM_SCP.2 *Problem tracking CM coverage*.

To meet all these configuration management assurance requirements, a configuration management plan needs to be designed and applied.

2. Delivery and Operation Assurance (Class ADO)

For basic and medium level security, EAL 2 and EAL 3 share the same assurance requirements from assurance Class ADO: ADO_DEL.1 *Delivery procedures* and ADO_IGS.1 *Installation, generation, and start-up procedures*. If high level security is needed in the development, EAL 4 should be imposed and it includes ADO_DEL.2 *Detection of modification* in addition to ADO_IGS.1.

In order to fulfill this class of security assurance requirements, documentation about the delivery and maintenance procedures and the installation, generation, and start-up of the TOE will be provided.

3. Development Assurance (Class ADV)

For basic level security, EAL 2 includes ADV_FSP.1 *Informal functional specification*, ADV_HLD.1 *Descriptive high-level design*, and ADV_RCR.1 *Informal correspondence demonstration*. For medium level security, EAL 3 requires ADV_HLD.2 *Security enforcing high-level design* as well as ADV_FSP.1 and ADV_RCR.1. As to high level security and EAL 4, more and higher level security assurance requirements from Class ADV are needed: 1) ADV_FSP.2 *Fully defined external interfaces*, 2) ADV_HLD.2 *Security enforcing high-level design*, 3) ADV_IMP.1 *Subset of the implementation of the TSF*, 4) ADV_LLD.1 *Descriptive low-level design*, 5) ADV_RCR.1 *Informal correspondence demonstration*, 6) ADV_SPM.1 *Informal TOE security policy model.*

A functional specification, high-level design, and analysis of the correspondence between the various TSF representations (e.g. TOE summary specification, functional specification, high-level design, low-level design, and implementation representation) is needed for the satisfaction of security assurance requirements of EAL 2, 3, and 4. Specifically for EAL 4, a subset of the implementation representation (e.g. source code), descriptive low-level design, and informal security policy model should also be presented.

4. Guidance Document Assurance (Class AGD)

For this assurance class, assurance requirements AGD_ADM.1 *Administrator guidance* and AGD_USR.1 *User guidance* need to be addressed. The administrator's manual and user's guide about the Grid TOE should be provided.

5. Life Cycle Support Assurance (Class ALC)

Consideration about assurance requirements from Class ALC is not necessary for basic level security and EAL 2. For medium level security, EAL 3 is imposed to include ALC_DVS.1 *Identification of security measures*, but for high level security, EAL 4 still includes two more requirements: ALC_LCD.1 *Developer defined life-cycle model* and ALC_TAT.1 *Well-defined development tools.*

To achieve EAL 3, all physical, procedural, personnel, and other security measures that are necessary to protect the confidentiality and integrity of the TOE design and implementation in its development environment should be identified. For achieving EAL 4, a life-cycle model and the development tools used for the TOE may be specified as well.

6. Tests Assurance (Class ATE)

Testing is an important method to provide assurance for the TOE development and implementation. For basic level security, EAL 2 is imposed and it includes following Tests related assurance requirements: ATE_COV.1 *Evidence of coverage*, ATE_FUN.1 *Functional testing*, and ATE_IND.2 *Independent testing – sample*. For medium level security and EAL 3, ATE_COV.2 *Analysis of coverage*, ATE_DPT.1 *Testing: high-level design* plus ATE_FUN.1 and ATE_IND.2 should be met. EAL 4 for high level security shares the same security assurance requirements with EAL 3.

As a result, an analysis of test coverage and/or depth of testing, a test plan which contains test procedures and test results, and sample independent testing should be made to satisfy the security assurance requirements of Class ATE.

7. Vulnerability Assessment Assurance (Class AVA)

EAL 2 for basic level security includes security assurance requirements AVA_SOF.1 *Strength of TOE security function evaluation* and AVA_VLA.1 *Developer vulnerability analysis*. For medium level security, EAL 3 requires one more assurance requirement than EAL 2: AVA_MSU.1 *Examination of guidance*. High level security and EAL 4 need higher assurance which can be achieved by fulfilling: 1) AVA_MSU.2 *Validation of analysis*, 2) AVA_SOF.1 *Strength of TOE security function evaluation*, and 3) AVA_VLA.2 *Independent vulnerability analysis*.

For EAL 2 and 3, the strengths of TOE security functions should be evaluated and the vulnerability analyzed. In case of EAL 4, the examination will also be provided for all the guidance and specification documents produced for the TOE.

5.1.3 Summary Specification Rationale

From the correspondence derived and demonstrated in Table 5.1, it can be seen that all of the CC disciplined IT security functional requirements for the Grid TOE are addressed by the specific TOE security functions summary defined in subsection 5.1.1. As shown in subsection 5.1.2, the stated TOE assurance measures are compliant with the Grid security assurance requirements.

TOE Security Functional Requirements/TOE Security Functions Summary	SF.Audit	SF.Authen_Comm	SF.Authorization	SF.AC_Interface	SF.Mgmt_DP
FAU_ARP	X			X	
FAU_GEN	X				
FAU_SAR	X		X		
FAU_SEL	X				X
FAU_STG	X				
FDP_ACC	X	X	X	X	X
FDP_ACF	X	X	X	X	X
FDP_DAU					X
FDP_ITC		X	X	X	X
FDP_RIP					X
FDP_SDI					X
FIA_AFL		X		X	
FIA_ATD		X	X		X
FIA_UAU		X			
FIA_UID		X			
FIA_USB		X	X		X
FMT_MOF			X		X
FMT_MSA			X		X
FMT_MTD	X		X		X
FMT_REV			X		X
FMT_SAE			X		X
FMT_SMF					X
FMT_SMR			X		X
FPR_ANO				X	X
FPT_AMT					X
FPT_FLS					X
FPT_RPL		X	X	X	
FPT_RVM				X	
FPT_STM				X	X
FPT_TDC					X
FTA_LSA				X	
FTA_MCS				X	
FTA_TAB				X	
FTA_TAH	X			X	
FTA_TSE				X	

Table 5.1 Security Functions Summary vs. Functional Requirements Mapping

5.2 Overview of a Specific TOE Implementation

In the modeling, derivation, design and implementation of the trustworthy TOE security functions for a specific computational Grid TOE (e.g. ClearingHouse) following the proposed Grid security assurance framework, most attention is paid to the security technical needs, functions and specifications which directly relate to the secure and dependable resource sharing and access at the Grid level. Functions which are indirectly related, such as those for hardware/software protection of the Grid resources at the resource host level and for resource management and scheduling, are currently left to the hardware/operating systems of the resource hosts and the Grid middleware.

For reasons of space, this thesis describes a specific implementation of only part of the complete Grid TOE that meets the Grid security functional and assurance requirements and the specific TOE summary specification drawn in sections 4.3 and 5.1. A TOE security function about authorization, TSF.Authorization, has been produced through a disciplined development process following the proposed Grid security assurance framework. Its design and implementation can be used as an example to demonstrate how a certain EAL is achieved in the disciplined development of a TOE security function, and further to show that this security engineering approach is also applicable for the development of other highly assured TOE security functions and the whole highly assured security functionality of a Grid TOE.

This sample TOE security function realizes a role based authorization for access control on Grid resources. The programs of TSF.Authorization implementation and testing are written, compiled and executed using Sun Java 1.5.0 [JAVA], and the authorization relevant information, such as user requests, resource policies and authorization decisions, is presented using the XACML (eXtensible Access Control Markup Language) [XACML]. Sun's XACML implementation library in Java [SXACML] is also utilized.

5.2.1 XACML

Because XML is fast becoming an industry standard with high compatibility and understandability, it can be used to specify authorization related user, resource and system information. One optional XML-based language is XACML. It targets the description of the context within which the user request is evaluated, and expresses resource policies upon which authorization decisions are made for information access over the internet. XACML even allows people to write a resource policy in terms of separate rules/sub-policies which can specify finer grained resource usage conditions individually, and the evaluation results of applying the rules/sub-policies respectively on the user request will be combined using a rule-combining/policy-combining algorithm to build a final authorization decision. This feature of XACML supports for more flexible specification, modification, and management of distributed resource policies and finer grained access control.

So far, the XACML schema does not explicitly add digital signatures, but common XML Digital Signature [XMLDSIG] solutions can be chosen to provide security for information passed over non-secured communication channels or paths. In fact, the authentication may even be performed by means of the XML Digital Signatures as well.

5.2.2 Functional Architecture of TSF.Authorization

The following scenario is implemented: a remote user wishing to utilize a remote Grid resource should authenticate himself to the resource's host or the resource's domain gatekeeper and then send his resource request for evaluation. The resource is protected and governed by a resource domain gatekeeper that can perform authentication and authorization operations. After the user is authenticated and his requested access granted, he can execute his requested action on the requested resource. A resource domain gatekeeper and the resource(s) governed by it constitute a resource domain, and there may be several resource domains in a Grid. For the sake of simplicity, the sample implementation demonstrated in this chapter focuses on authorization and assumes that the user has already been successfully authenticated and can communicate with the resource domain gatekeeper securely using SSL [SSL]. The authentication mechanism can be a TOE security function about authentication implemented (for example, utilizing GSI libraries) in the same resource domain gatekeeper and meeting the Grid security functional requirements from Functional Class FIA (Identification and Authentication), but it is not included in the current implementation package of TSF.Authorization. Secure data storage is also left to the relevant mechanisms implemented to meet its associated Grid security functional requirements in Functional Classes FDP (User Data Protection), FMT (Security Management) and FPR (Privacy).

In this experimental implementation, the relevant information handled by TSF.Authorization is encoded in XACML and these XACML files can be proved trustworthy by their XML digital signatures which were appended by an attribute or policy authority or created using some trusted private keys, but it is assumed that the contents of these XACML files are trusted as long as they are transmitted securely between authenticated parties. It should be noted that although the libraries about generating XACML format files and about producing/verifying digital signatures will not be introduced in detail in the following assurance documentation about TSF.Authorization, the former has already been incorporated in the current implementation package and the latter can be made easily by utilizing existing security libraries.

The architecture of the implementation package of TSF.Authorization includes three main components: 1) RemoteClient, 2) RemoteResourceGatekeeper, and 3) PolicyServer. The RemoteClient is located on the remote user's computer and sends access requests for the remote resource on behalf of an authenticated user. The RemoteResourceGatekeeper, as the implementation of a resource domain gatekeeper, governs access to the (one or more) remote resources in its resource domain and accepts remote users' requests. It evaluates the request against resource policy/policies (there may be more than one access policy for a single resource because the constraints on the access to that resource could be distributed in multiple policy files to achieve more flexibility in the application), and returns an evaluation result to the remote user. It relieves individual resource hosts from abundant computations incurred by evaluating each incoming user request, and allows the resource power itself to be used more for serving user jobs rather than used for the authorization. The RemoteResourceGatekeeper needs to be assured by the access control function of the TOE (or environment thereof) to be the only way of obtaining access to the remote Grid resources. It retrieves trusted resource policy/policies from a third party such as the PolicyServer and thus reduces the needs of individual resource hosts to store policies because a shared policy base can serve a variety of distributed Grid resources flexibly. It is also easier to maintain and protect a single policy server than policies stored on globally dispersed resource hosts. In the implementation, authorization is not simply identity based as widely adopted by other Grid security solutions. Because of the role based authorization, the evaluation for the user should not only be based on his identity, but also on his role/attributes and other possible environmental factors. All such role/attributes constraints about accessing a Grid resource are encoded and reflected in relevant XACML policy files. The requesting user shall provide his role/attributes relevant information in his XACML formatted resource request, and the content of the request will be interpreted in the RemoteResourceGatekeeper. Although it is a functional issue at the implementation level and thus is not compulsory for all specific Grid TOEs developed following the Grid security assurance framework, the role based authorization, rather than the identity based and coarse grained authorization, can provide a more flexible, finer grained and more powerful access control for the Grid and a better protection for Grid resources. By adopting such a systematically developed role based authorization function for the Grid, the efficiency of securing the Grid systems/applications will be improved while the high assurance is achieved at the same time.

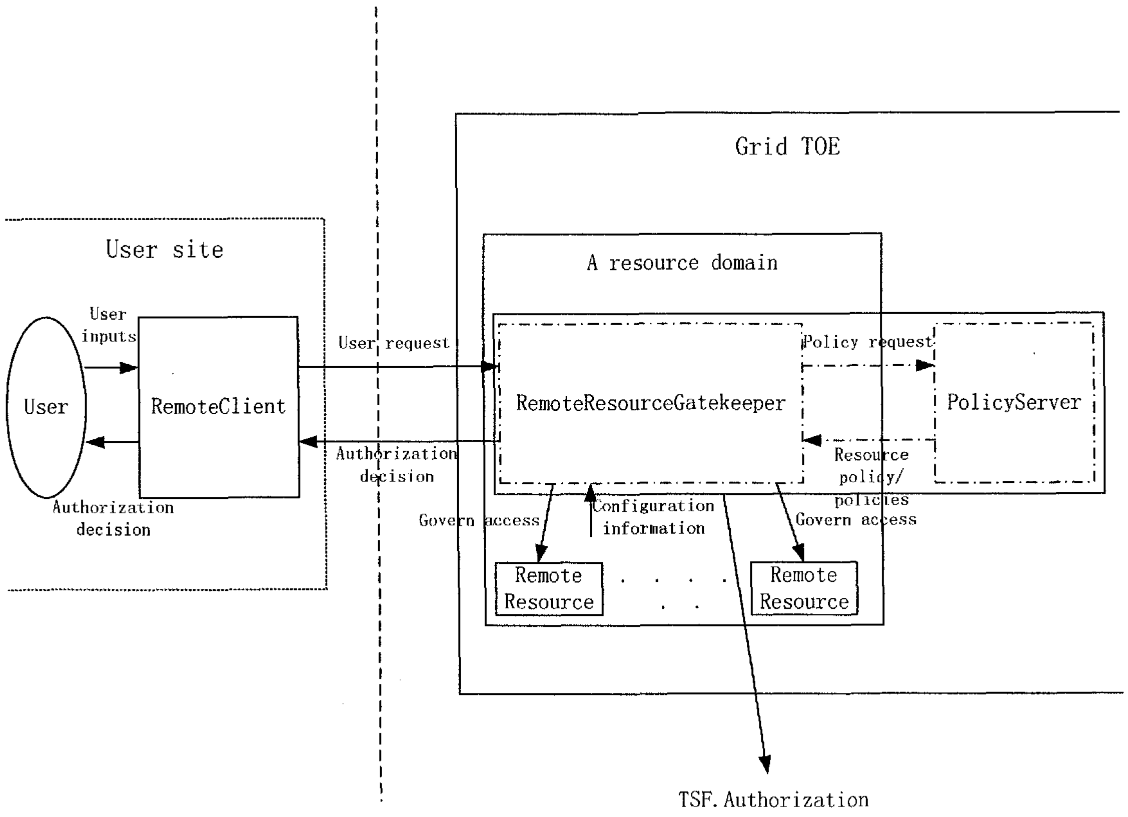

Figure 5.1 TSF.Authorization for the Grid TOE

Figure 5.1 shows that the components of the implementation package belong to two realms - the Grid TOE and the User site. The RemoteResourceGatekeeper and the

PolicyServer are two TSF.Authorization components in the Grid TOE, and the core evaluation work of the authorization function will be done in the RemoteResourceGatekeeper. As for PolicyServer host, it provides storage and collection of role based resource access policies. The RemoteResourceGatekeeper and PolicyServer work together to fulfill TSF.Authorization, but the PolicyServer was designed to provide better flexibility and efficiency for the authorization in practice, and it does not directly address authorization related Grid security functional requirements. Possible authentication and secure communication between the RemoteResourceGatekeeper host and the PolicyServer host, like those between the RemoteClient and TSF.Authorization, are left to the underlying infrastructure, and the PolicyServer need not be implemented if resource policies are stored locally. In the Grid TOE, there can be several resource domains, each of which consists of one RemoteResourceGatekeeper, one or more Grid resources, and also one or more PolicyServers which may be shared among several resource domains. A RemoteClient component is included in the implementation package because computer facilities are needed for the human user to find the appropriate RemoteResourceGatekeeper, send the user's resource request to be evaluated, and receive and display the evaluation result of authorization. It is also useful for the visualization of the operation and testing of the system. The RemoteClient and the human user himself are both within the User site, a constituent of the TOE environment (i.e. not part of the TOE), and can interact with the Grid TOE and Grid resources.

5.2.3 Achieving Assurance for the Implementation

For the specific implementation of TSF.Authorization, EAL 3, which is suitable for medium level security, is chosen as the extent of security assurance to be achieved by the TOE. As shown in subsection 5.1.2, EAL 3 requires assurance evidence that the stated assurance measures are enforced to satisfy the relevant Grid security assurance requirements on configuration management, delivery and operation, development, guidance documents, life cycle support, tests, and vulnerability in the implementation. In the following sections, assurance documents will be provided to demonstrate how the required assurance is gained from the enforcement of development, life cycle support, and tests relevant assurance measures for TSF.Authorization. These documents were produced together with a systematic implementation process following the Grid security assurance framework. Assurance evidence for the assurance measures and requirements related to other assurance classes would be given at the end of the development of the complete Grid TOE.

5.3 Development Assurance

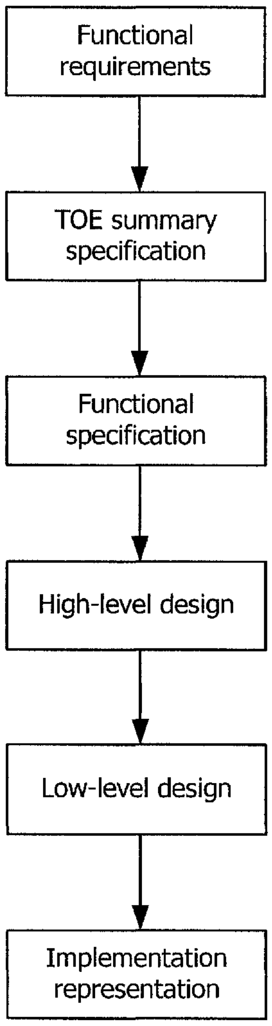

Figure 5.2 Relationships between Different Grid TOE Representations

The process from specifying a TOE summary to writing the final program code is also a process of representing the TOE security functions at various levels of abstraction, from the highest level abstract specification to the lowest level detailed implementation representation.

As shown in Figure 5.2, in the design and implementation stage of the specific Grid TOE's security engineering development process, the TOE summary specification will first be expanded and elaborated to a functional specification. The functional specification is a high-level description of the user-visible interface and behavior of the TSF. It is an instantiation of the TOE security functional requirements. In this section, only functional specification for TSF.Authorization is presented while the implementation of other TOE security functions (such as those about audit (TSF.Audit), authentication (TSF.Authentication&SecureCommunication), access control (TSF.AccessControl), and security management and data protection (TSF.SecurityManagement&DataProtection)) may also contribute to the effective functioning of TSF.Authorization. Next, the functional specification will be extended to make a high-level design of the TSF. The high-level design provides a description of the TSF in terms of major structural units (i.e. subsystems) with their purposes and interfaces, and relates these structural units to the functions that they provide. The production of a correct high-level design will bring assurance that the architecture is appropriate to implement the security functional requirements. By refining and elaborating the high-level design of the TSF, a low-level design can be obtained as a detailed description of the internal working of the TSF and TSF subsystems in terms of finer grained and lower level functional modules and their interrelationships and dependencies. The low-level design provides assurance that the TSF subsystems have been correctly and effectively refined. Finally, the representation of the implementation is in the form of source code, firmware, hardware, etc. and captures the most detailed and specific internal workings of the TSF. In most Grid security development practices, the implementation representation is the main TOE and TSF form that people care about, and the assurance measures for more abstract representations of the TOE are considered less in the development. In contrast, the Grid security assurance framework proposed in this thesis addresses all levels of the TOE representation. Since EAL 3 is imposed on the current implementation, only the consistent functional specification and high-level design are needed for the assurance evidence on the fulfillment of the corresponding development assurance requirements. In view of the thesis length, detailed descriptions of the external interfaces of TSF.Authorization (for the functional specification) and the internal interfaces between subsystems of TSF.Authorization (for the high-level design) are not included in this section.

5.3.1 Functional Specification

According to ADV_FSP.1 *Informal functional specification*, this functional specification covers the Grid security functional requirements related to authorization (such as FDP_ACC *Access control policy* and FDP_ACF *Access control functions*) and the relevant security functions defined in the Grid TOE's summary specification. In this implementation, the role based authorization function is realized by the RemoteResourceGatekeeper and the PolicyServer and a remote Grid user is represented by the RemoteClient interacting with TSF.Authorization, with the support and coordination of the required underlying hardware, software and middleware.

The PolicyServer should be started first to wait for any retrieval request for a specific resource access policy. Upon start-up, the RemoteResourceGatekeeper must be configured with the essential information (e.g. location) of the PolicyServer(s) which it may contact when evaluating users' requests. The configuration can use some pre-embedded parameters or run-time inputs from the administrator.

The running TSF.Authorization (including both the PolicyServer and the RemoteResourceGatekeeper) can accept access requests from the RemoteClient for the Grid resources that the TSF governs. In the current implementation, there are two types of people that may interact with TSF.Authorization. One is the user who is a remote S.CommonUser and handled by the authorization function, and the other is the administrator who belongs to a subset of S.Admins and may only manage the functioning of the RemoteResourceGatekeeper and PolicyServer. They communicate with the RemoteResourceGatekeeper via different interfaces. After the RemoteClient is started, the user should configure the RemoteClient with the location of his preferred RemoteResourceGatekeeper. The location information of the RemoteResourceGatekeeper that governs access to the user's target resource can be found from relevant public Grid information services in advance. A specific resource access request will then be input to the RemoteClient and forwarded to the RemoteResourceGatekeeper. The resource request includes the user's personal information (such as the name and the role/attributes) needed for authorization decisions, the name of the target resource, and the action that the user wishes to perform.

Annex A presents a sample resource request.

5.3.1.1 TSF.Authorization Summary

Upon receiving an XACML formatted resource access request from the RemoteClient, the RemoteResourceGatekeeper checks which resource is requested and generates a possible policy request to retrieve the resource's access policy/policies from the

PolicyServer, and the PolicyServer information provided when the RemoteResourceGatekeeper was started is utilized to help send the policy request to the appropriate PolicyServer.

When receiving the policy request, the PolicyServer interprets the policy request and seeks a policy base for resource policy/policies. An XACML formatted resource policy includes a RuleCombiningAlgId attribute to identify the rule combining algorithm it adopts, and a Target component to describe which users, resources, and actions are applicable to this policy. Besides, one or more policy-specific Rule elements are also presented concerning the permitted/denied user identities/roles/attributes, permitted/denied actions on the resource, and/or other possible conditions that are employed by the rule. A sample resource access policy is shown in Annex B.

The searching result of PolicyServer is sent to the requesting RemoteResourceGatekeeper for use by the evaluation routine, and the interactions between the RemoteResourceGatekeeper and the PolicyServer are internal to TSF.Authorization and invisible outside.

After collecting the user's resource request and related resource policy/policies, the RemoteResourceGatekeeper performs authorization operations by evaluating the user request against the policy/policies. The remote user's role/attributes, requested action, and possible environmental factors should strictly satisfy the constraints and conditions contained in the role based resource access policy/policies for the user to gain access. Such an implementation satisfies FDP_ACC.1 *Subset access control* and FDP_ACF.1 *Security attribute based access control* because the users, resources and actions confined in FDP_ACC.1 are all considered to be handled by TSF.Authorization, and the access control is exactly based on the requesting user's role/attributes as stated in FDP_ACF.1. A user's id/role/attributes supported by XACML include subject-id, subject-category, key-info, request-time, and any possible string format attribute which is customized and recognized by the corresponding XACML formatted resource policies. The values of user role/attributes are resource-specific, and different values may be assigned to the same user role/attribute to be used for accessing different Grid resources. For example, a remote user can have an Alumni role according to resource A and a Postgraduate Student role which is only applicable to resource B. Although there are some user role/attributes whose values are widely accepted by many resources, the values are still void when outside the circle of those specific resources so that such role/attribute values are none the less resource(s)-specific. These user role/attributes are similar to an X.509 identity certificate whose content (value) is trusted by any resource that trusts the certificate's CA but meaningless to other resources. In such situations, different resource-specific user role/attribute values may need to be selected for the user's requests to different resources.

For a resource policy, it can contain more than one Rule element each of which is used to match with the corresponding elements in the user's resource request independently. Consequently, contradictions may appear among the results of assessing the resource request using different rules of a policy. For instance, by Rule 1 we may conclude that the user's requested action gets "Permit" while by Rule 2 the result is "Deny". To solve such problems, a rule combining algorithm should be stated in each policy to govern the combination of the results of applying individual rules of this policy. A policy author can adopt a standard rule combining algorithm provided by XACML: "Deny-overrides", "Permit-overrides", or "First applicable" (which means the evaluation result against the first rule in the policy will override the evaluation results against other rules in this policy) or even customize a new combining algorithm compatible with the XACML specifications. The RemoteResourceGatekeeper will make use of the policy-specific rule combining algorithm referenced by the RuleCombiningAlgId attribute in the resource policy to generate the final evaluation result against this policy.

If there is only one role based access policy collected for the requested Grid resource, the evaluation result using this policy will be the final result of the whole evaluation. However, it is also possible that a resource has several distributed stakeholders to define policies for it. Each stakeholder may create his own resource policy file and store it on the PolicyServer so that there is more than one access policy (represented by more than one policy file) for a single resource. Alternatively, even though the Grid resource has only one policy author, he may also want to express the resource policy in several policy files to achieve more flexibility. For such cases, a policy combining algorithm is needed as well to combine all the results of evaluating the user's resource request against the associated resource policy files individually to solve the possible contradictions among them. The RemoteResourceGatekeeper will use the "Deny-overrides" policy combining algorithm in the authorization.

Considering the details of evaluation, Subject, Resource, and Action elements of the user's resource request should first be matched with the corresponding elements in the Target component of the policy, using the matching methods which are also stated in the corresponding elements. If the attribute values of an element in the policy's Target component are not a subset of the attribute values of the corresponding element in the user's resource request, then the policy's Target component is not matched by the request and the policy is not applicable to this request, and a "Not Applicable" decision will thus be generated. For instance, the Subject element of the resource request in Annex A can be accepted by the Target component of a policy used to evaluate the request only when in the Subjects element of this Target component, there is

- a Subject element that only has the attribute and attribute value: "subject-id – O=Grid, OU=NTUCG, OU=NTUCG-ntuca.ntu.edu.sg, OU=ntu.edu.sg, CN=Zhu Ning"; or
- a Subject element that only has the attribute and attribute value: "role – NTU postgraduate student"; or
- a Subject element that only has the attributes and attribute values: "subject-id – O=Grid, OU=NTUCG, OU=NTUCG-ntuca.ntu.edu.sg, OU=ntu.edu.sg, CN=Zhu Ning" and "role – NTU postgraduate student"; or
- only an AnySubject element (equaling a Subject element with null attribute and attribute value).

Since the Subjects, Resources, and Actions elements of the Target component of the policy shown in Annex B only contain AnySubject, AnyResource, and AnyAction elements respectively, this policy is applicable for evaluating any remote user's resource requests.

If the resource policy is verified to be applicable for further evaluating the current user resource request, then the request has to be evaluated against each Rule of the policy. Similarly, the elements of the request are matched with the corresponding elements in the Rule following the matching methods referred to by the MatchIDs of these corresponding elements. In addition, since the resource stakeholder/policy author may add other constraints on accessing the resource such as conditions about the environmental factors, the possible Condition elements in the Rule should also be matched with the real time environment in a manner stated by FunctionId. For example, if there is a condition that requires the accessing time to be between 9am and 5pm Singapore Time daily, then only the user requests received in this period will be accepted by the condition. If all relevant elements in a Rule are matched by the remote user's resource request and the environment, then the value of the Effect attribute (Permit/Deny) of this Rule will be output. If not, then a "Not Applicable" will be output. When the request has been evaluated against all Rules of the policy, all the results will be combined following the algorithm stated by the RuleCombiningAlgId attribute of the policy to build the evaluation result against this policy.

In case that 1) there is only one access policy for the resource, then the evaluation result against this policy is the final result of the evaluation routine for the remote user's resource request; 2) there is more than one access policy for the resource, then the final result should be the combination of the results of individually evaluating the request against each policy under the enforcement of the policy combining algorithm; 3) there is no access policy for the resource, then a "Not Applicable" result will end the whole evaluation.

To sum up, there are four decisions that may be made by the evaluation routine:

1) "Permit" – when
    a) there is at least one resource policy applicable to the resource request and no policy against which the evaluation result is "Deny"; and
    b) in that policy, there is at least one satisfied rule whose Effect is "Permit"; and
    c) in that policy, there is no satisfied rule whose Effect is "Deny", or the rule combining algorithm is "Permit-overrides", or the satisfied rule whose Effect is "Permit" is the first rule in the policy and the rule combining algorithm is "First applicable".

2) "Deny" – when
   a) there is at least one resource policy applicable to the resource request; and
   b) in that policy, there is at least one satisfied rule whose Effect is "Deny"; and
   c) in that policy, there is no satisfied rule whose Effect is "Permit", or the rule combining algorithm is "Deny-overrides", or the satisfied rule whose Effect is "Deny" is the first rule in the policy and the rule combining algorithm is "First applicable".
3) "Not Applicable" – when no resource policy used in the evaluation is applicable or no access policy can be found for the requested resource, when the value of the resource-id attribute in the remote user's resource request does not equal the name of any resource governed by the current RemoteResourceGatekeeper, or when the value of the resource-id attribute contained in the actually retrieved resource access policy does not equal the value of the resource-id attribute included in the corresponding policy request which was generated and sent by the RemoteResourceGatekeeper. A resource policy is not applicable to the remote user's resource request if it follows any one of below:
   a) the Target component is not applicable to the resource request;
   b) the evaluation result against any rule in this policy is always "Not Applicable";
   c) there is an evaluation result of "Not Applicable" against the first rule in the policy and the rule combining algorithm is "First applicable".
4) "Indeterminate" – when some error occurs so that the requested access is unable to be evaluated. Reasons for such inability include: network errors while retrieving policies, division by zero during policy evaluation, syntax errors in the resource request or resource policy, etc.

In a policy, if the rule combining algorithm is "Permit overrides" (or "Deny overrides") and there are only one or more rules against which the evaluation results are "Deny" (or "Permit") plus zero or more rules against which the evaluation results are "Not Applicable", then the evaluation result against this policy should be "Deny" (or "Permit"). If there are only inapplicable rules in the policy, the evaluation result against this policy will be "Not Applicable" whatever the rule combining algorithm. These considerations also apply to the similar situations in combining the results of evaluating the user's resource request against a set of access policies for a single Grid resource.

The final authorization decision from the RemoteResourceGatekeeper will be encoded in XACML and sent back to the RemoteClient for notifying the remote user and any follow-on operations. Only when a "Permit" is given to the requesting user can he perform his requested action on the requested resource. The result of evaluating the resource request in Annex A against the resource policy in Annex B at a time between 09:00 and 17:00 GMT+8 is "Permit" and the details are shown in Annex C as an example.

5.3.1.2 TSF.Authorization Interfaces

1. **Interface to the Administrator**
   - AuthorizationTSF::Configuration (location(s) of PolicyServer(s))
     - Load the configuration data for the RemoteResourceGatekeeper

2. **Interface to the RemoteClient**
   - AuthorizationTSF::acceptConnection (incoming connection from the RemoteClient, user request)
     - Connect with the end user's RemoteClient;
     - Receive the remote user's resource request for authorization;

- o Return the authorization decision

5.3.2 High-Level Design

The high-level design is refined from the functional specification for TSF.Authorization and fulfills the assurance evidence needs as stated in ADV_HLD.2 *Security enforcing high-level design.* In the specification of this part, RemoteClient, RemoteResourceGatekeeper, and PolicyServer are defined as three separate system components simply because they will respectively be located and work on three different hosts in our implementation for TSF.Authorization. The division into such components is simply for more convenient and clearer specification and expression of the design and implementation, and therefore is not necessary for a TOE security function. The subsystems within a component are the structural constituents of TSF.Authorization at this level of abstraction and there may be one or more subsystems located on a component's host.

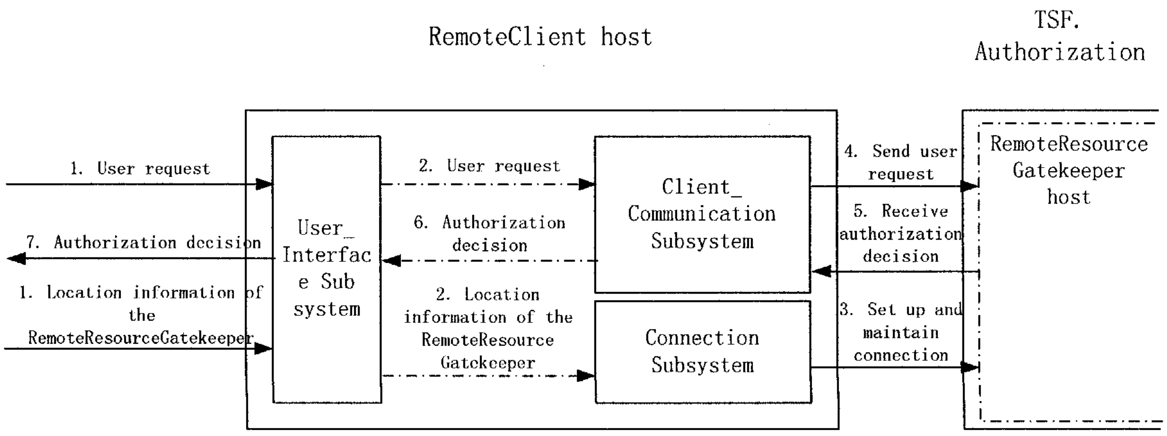
RemoteClient host
TSF.
Authorization
1. User request
7. Authorization decision
1. Location information of the RemoteResourceGatekeeper
User_ Interface Sub system
2. User request
6. Authorization decision
2. Location information of the RemoteResource Gatekeeper
Client_ Communication Subsystem
Connection Subsystem
4. Send user request
5. Receive authorization decision
3. Set up and maintain connection
RemoteResource Gatekeeper host

Figure 5.3 Structure of the RemoteClient

The RemoteClient includes a User_Interface subsystem, which forwards the user's request and reads in the location information of the target RemoteResourceGatekeeper. The final authorization decision and possible error messages are displayed to the user also via this subsystem. The Connection subsystem is used to maintain a connection with the RemoteResourceGatekeeper, and the Client_Communication subsystem is for sending the user request and receiving the authorization decision. In the division of duties between the Connection subsystem and the Client_Communication subsystem, the former handles the application level protocol and connection whereas the latter handles the information exchange over the connection. The work flow of the RemoteClient is illustrated in Figure 5.3.

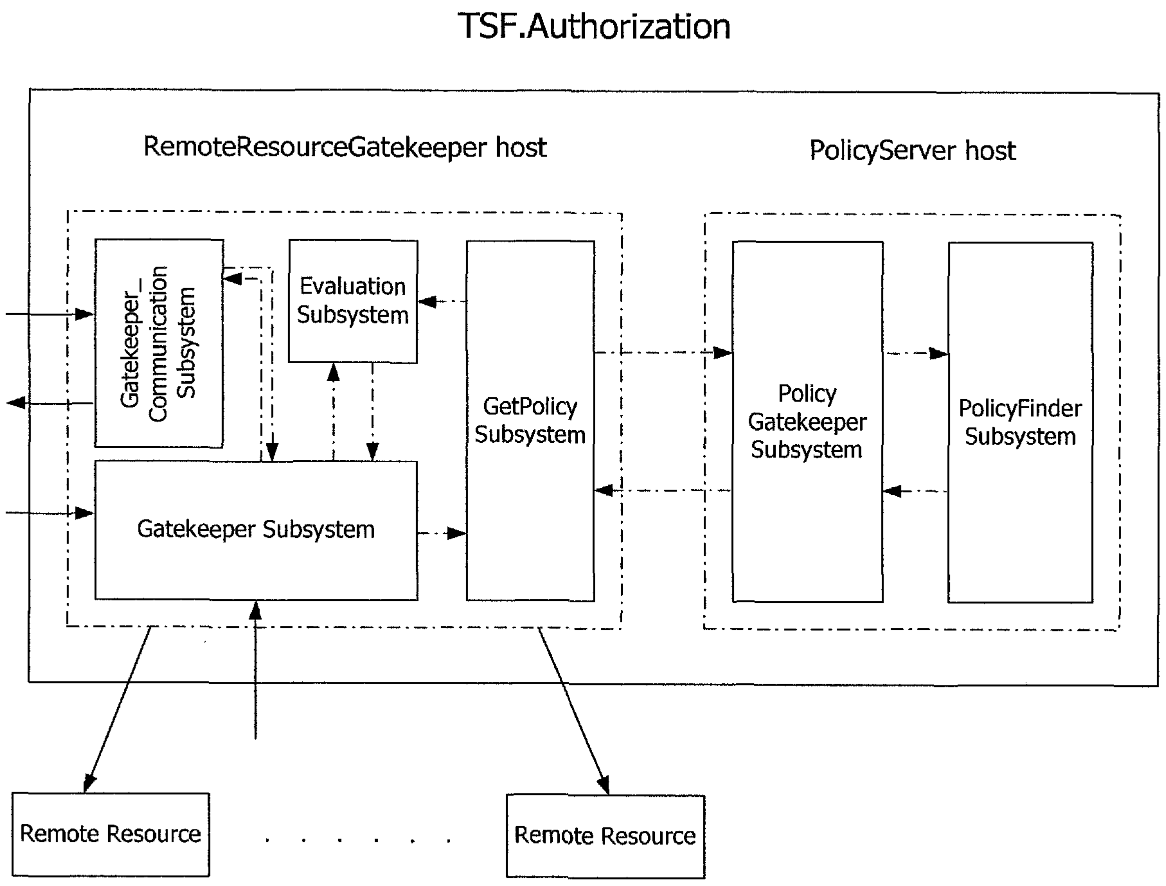

Figure 5.4 Composition of TSF.Authorization

Figure 5.4 provides a view of the various subsystems of TSF.Authorization. The subsystems are designed to be distributed on two separate machines to perform relatively distinct functions constituting TSF.Authorization. Among them, the Gatekeeper, Gatekeeper_Communication, Evaluation, and GetPolicy subsystems reside in the

RemoteResourceGatekeeper host, and the PolicyGatekeeper and PolicyFinder subsystems are on the PolicyServer host.

5.3.2.1 RemoteResourceGatekeeper

The RemoteResourceGatekeeper 1) accepts connections from the RemoteClients; 2) receives and interprets remote users' resource requests; 3) collects role based access policies for specific resources; and 4) evaluates user requests against the resource policies to determine the evaluation results of authorization.

In the RemoteResourceGatekeeper, the Gatekeeper subsystem is responsible for reading in the configuration information of the RemoteResourceGatekeeper, for accepting incoming connections from RemoteClients, for generating policy requests based on user requests to retrieve access polices of the requested Grid resources, and for passing user requests, policy requests, and authorization decisions to the Evaluation subsystem, the GetPolicy subsystem, and the Gatekeeper_Communication subsystem respectively. The Gatekeeper_Communication subsystem reads in the remote user's resource request from RemoteClient via the connection between the RemoteResourceGatekeeper and the RemoteClient, and sends the authorization decision back to the RemoteClient. As for the GetPolicy subsystem, it locates the relevant PolicyServer and retrieves resource access policies for sending them to the Evaluation subsystem. The Evaluation subsystem is used to assess the remote user's resource request against the resource policy/policies to determine authorization. The user identity and role/attributes, requested action, and possible environmental factors are parsed, interpreted and matched with the corresponding elements in the resource policy/policies to conclude the evaluation result: "Permit", "Deny", "Not Applicable", or "Indeterminate".

Figure 5.5 illustrates the architecture of the RemoteResourceGatekeeper as well as the handling process for a user's resource request.

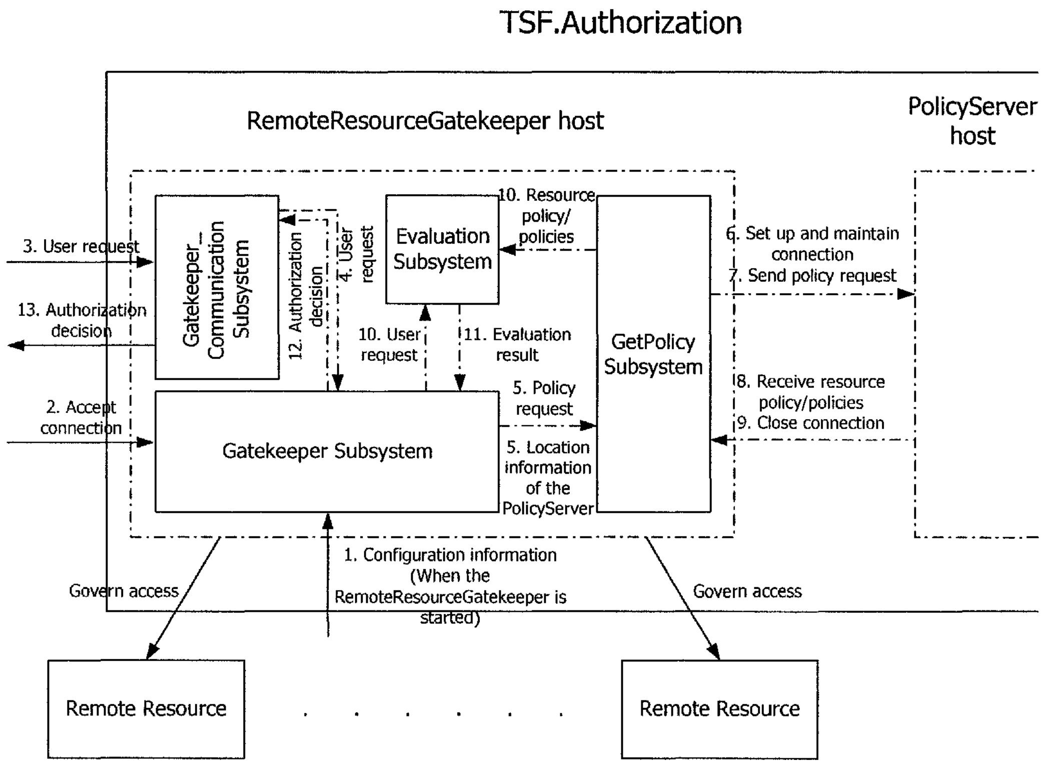
TSF.Authorization
RemoteResourceGatekeeper host
PolicyServer host
Gatekeeper_ Communication Subsystem
Evaluation Subsystem
GetPolicy Subsystem
Gatekeeper Subsystem
3. User request
13. Authorization decision
2. Accept connection
12. Authorization decision
4. User request
10. Resource policy/ policies
10. User request
11. Evaluation result
5. Policy request
5. Location information of the PolicyServer
6. Set up and maintain connection
7. Send policy request
8. Receive resource policy/policies
9. Close connection
1. Configuration information (When the RemoteResourceGatekeeper is started)
Govern access
Govern access
Remote Resource
Remote Resource

Figure 5.5 Subsystems on the RemoteResourceGatekeeper Host

The subsystems and their interfaces in the RemoteResourceGatekeeper are as follows:

1. Gatekeeper Subsystem Interfaces

- Interface to the Administrator: Gatekeeper::Configuration (location(s) of PolicyServer(s))
  - o Load the configuration data for the RemoteResourceGatekeeper when it is started
- Interface to the RemoteClient: Gatekeeper::acceptIncomingConnection (incoming connection from the RemoteClient)
  - o Accept and maintain incoming connections initiated by RemoteClients
- Interface to the Gatekeeper_Communication Subsystem: Gatekeeper:: InternalCommunication (user request)

- Receive the user's resource request from the Gatekeeper_Communication subsystem for further handling
- Send the authorization decision to the Gatekeeper_Communication subsystem to be forwarded to the RemoteClient

2. Gatekeeper_Communication Subsystem Interfaces

- Interface to the RemoteClient: Gatekeeper_Communication:: ExternalCommunication (user request)
  - Receive the resource request sent by the RemoteClient on behalf on a remote user
  - Return the final authorization decision to the RemoteClient

3. GetPolicy Subsystem Interfaces

- Interface to the Gatekeeper Subsystem: GetPolicy::retrievePolicy (policy request, location of PolicyServer)
  - Receive the generated policy request from the Gatekeeper subsystem for retrieving the access policy/policies of the requested Grid resource from the PolicyServer

4. Evaluation Subsystem Interfaces

- Interface to the Gatekeeper Subsystem: Evaluation::Eva (user request)
  - Receive the user's resource request from the Gatekeeper subsystem for accessing the user request against the role based access policy/policies of the requested resource
  - Return an evaluation result to the Gatekeeper subsystem for the remote user's authorization

- Interface to the GetPolicy Subsystem: Evaluation::EvaPolicy (resource policy/policies)
    - o Read in the retrieved role based access policy/policies of the requested Grid resource from the GetPolicy subsystem to be used for evaluating the remote user's resource request

5.3.2.2 PolicyServer

The PolicyServer host provides a centralized storage of resource access policies separate from specific resource hosts. It accepts incoming connections and policy requests from the RemoteResourceGatekeeper host, finds access policies for the requested resources, and returns the policies to the RemoteResourceGatekeeper. The PolicyServer consists of a PolicyGatekeeper subsystem to handle connections with RemoteResourceGatekeepers, accept policy requests and finally send back the requested resource policy/policies or other possible responses, and a PolicyFinder subsystem to locate appropriate access policy/policies for the requested Grid resource and return the result of searching to the PolicyGatekeeper subsystem. These are illustrated in Figure 5.6.

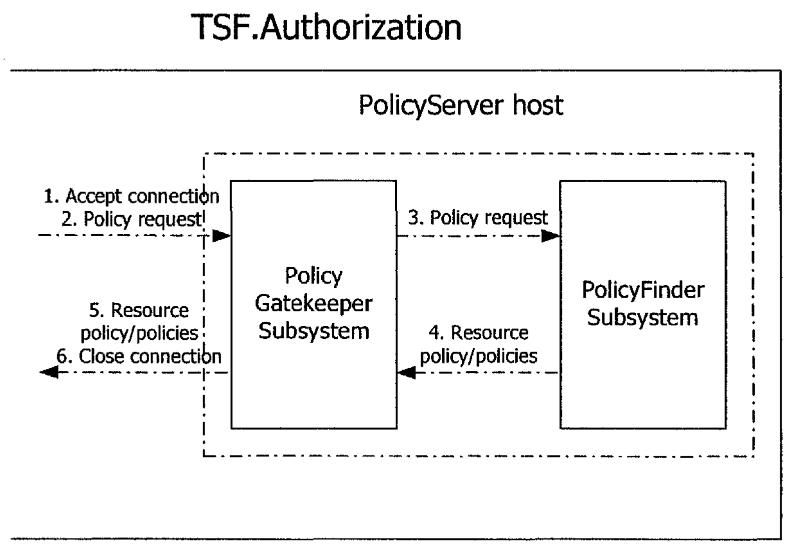

Figure 5.6 Subsystems on the PolicyServer Host

The subsystems and their interfaces in the PolicyServer are as follows:

1. **PolicyGatekeeper Subsystem Interfaces**
   - Interface to the RemoteResourceGatekeeper host (GetPolicy Subsystem): PolicyGatekeeper::acceptPolicyRequest (incoming connection from the RemoteResourceGatekeeper, policy request)
     - o Receive the policy request from the GetPolicy subsystem on the RemoteResourceGatekeeper host and pass it to the PolicyFinder subsystem for finding the requested resource access policy/policies
     - o Return the found resource access policy/policies or other possible responses to the GetPolicy subsystem on the RemoteResourceGatekeeper host

2. **PolicyFinder Subsystem Interfaces**
   - Interface to the PolicyGatekeeper Subsystem: PolicyFinder::handlePolicyRequest (policy request)
     - o Receive and interpret the policy request to find the appropriate resource access policy/policies
     - o Send the result of searching to the PolicyGatekeeper subsystem

5.3.3 Representation Correspondence

The rationale in this part is to demonstrate that the Grid security assurance requirement ADV_RCR.1 *Informal correspondence demonstration* is met.

5.3.3.1 TOE Summary Specification to Functional Specification Correspondence

For our TSF.Authorization implementation for the Grid TOE, only the tracings between TSF.Authorization and the TOE summary specification need to be considered, but the other security functions of the TOE may also contribute to or complement the correct and effective functioning of the whole authorization security function. For example, when a remote user's resource request passes the evaluation performed by the authorization security function, the access control function of the TOE as well as other relevant mechanisms of the TOE and the user's target resource host should provide the user with the successful delegation of the relevant local privileges and the proper pathway for accessing and operating on the resource. Also the user's personal and job data needs to be managed and protected by the TOE's appropriate security management and data protection functions. All of the assumptions, underlying support, and characteristics of the TSF.Authorization implementation should also be reflected in other relevant assurance evidence such as that for the Assurance Class AGD (Guidance Documents).

TOE Summary Specification/ Functional Specification	TSF. Audit	TSF. Authentication&Secure Communication	**TSF. Authorization**	TSF. Access Control	TSF.Security Management& DataProtection
SF.Audit	X				X
SF.Authen_Comm	X	X		X	X
SF.Authorization	X		**X**	X	X
SF.AC_Interface	X	X	**X**	X	X
SF.Mgmt_DP	X	X	**X**		X

Table 5.2 TOE Summary Specification to Functional Specification Correspondence

Consequently, in Table 5.2, an additional correspondence is also given to address the tracings that may exist from those further non-authorization TOE security functions to the TOE summary specification.

5.3.3.2 Functional Specification to High-Level Design Correspondence

In Table 5.3, the correspondence between the functional specification of TSF.Authorization and its high-level design is demonstrated. For the reason that the design of TSF.Authorization can also contribute to the functioning of the other TOE security functions, the tracings between TSF.Authorization's high-level design and the other TOE security functions are included as well.

High-Level Design/Functional Specification	TSF. Audit	TSF. Authentication& Secure Communication	**TSF. Authorization**	TSF. Access Control	TSF.Security Management& DataProtection
Gatekeeper Subsystem	X	X	X	X	X
Gatekeeper_ Communication Subsystem			X		
Evaluation Subsystem			X		X
GetPolicy Subsystem	X		X		
PolicyGatekeeper Subsystem	X		X		
PolicyFinder Subsystem			X		X

Table 5.3 Functional Specification to High-Level Design Correspondence

5.4 Life Cycle Support Assurance

Life cycle support is an aspect of establishing discipline and control during the development, maintenance, refinement and evolvement of the TOE. In our development and maintenance, security analysis and the production of the assurance evidence according to EAL 3 are performed on a regular basis, as an integral part of the development and maintenance activities. Greater confidence in the correspondence among the TOE security requirements and the different TOE representations is thus obtained.

5.4.1 Development Security

In this subsection, the Grid security assurance requirement ALC_DVS.1 *Identification of security measures* is addressed for life cycle security and achieving the EAL 3 assurance.

In our development, the physical environment of the TOE design and implementation is protected so that only legal developers, supervisors, managers, and investigators can access it. All outdated development relevant information has been securely kept or destroyed using trusted facilities and by authorized people. The people involved in the design and implementation are competent professionals and researchers with required techniques and knowledge. Finally, a check has been made to ensure that all the development security relevant security measures are followed during the development and maintenance.

5.5 Tests Assurance

The testing following assurance Class ATE (Tests) is a means to ensure that the TOE security functional requirements are correctly and effectively met. It establishes that the implemented TSF.Authorization works according to the security specifications and designs, i.e. it must exhibit correct behaviors when called through the usual interfaces, and undesirable behaviors must be absent. In this section, assurance evidence at EAL 3 is provided for satisfying the Grid security assurance requirements stated in ATE.FUN.1 *Functional testing*, ATE_COV.2 *Analysis of coverage*, and ATE_DPT.1 *Testing: high-level design*. The fulfillment of the assurance requirement ATE_IND.2 *Independent testing – sample* is left to the evaluators of TSF.Authorization.

5.5.1 Functional Testing

1. Test environment

- Three computers are involved to deploy the implementation package in the testing. They are interconnected via the Nanyang Campus Grid. The first computer is used to host the RemoteClient, the second to host the RemoteResourceGatekeeper component of TSF.Authorization, and the third to host TSF.Authorization's PolicyServer component and to provide centralized storage of resource access policies.

- Hardware: The RemoteClient host is a laptop with Intel P4-M CPU 1.60GHz and 512MB RAM, the RemoteResourceGatekeeper host is a PC equipped with Intel P4 CPU 2.66GHz and 512MB RAM, and the PolicyServer host is a PC with Intel P4 CPU 2.20GHz and 1GB RAM.
- Software: The RemoteClient host and the PolicyServer host are both running Microsoft Windows XP Professional SP2, and the RemoteResourceGatekeeper host is running Microsoft Windows 2000 Professional SP4.

2. Test conditions

- It is required that the possible firewall of each machine involved in the tests is configured to permit the authorization relevant interactions between the machines.
- Authentication and secure communication/data storage are assumed to be fulfilled by the underlying infrastructure (but can be added to the implementation package) and are transparent to the testers in the current TSF.Authorization implementation and testing.
- The resource request sent by the RemoteClient to TSF.Authorization is identical with that sent by the requesting user to the RemoteClient, the location of the RemoteResourceGatekeeper with which the RemoteClient requests connection is the same as that provided by the user to the RemoteClient, and the evaluation result of authorization received by the RemoteClient from TSF.Authorization is the same as that displayed to the user by the RemoteClient (These have been verified by adding relevant code to the RemoteClient programs to print out the relevant data for comparison, but this is testing on the RemoteClient, not on TSF.Authorization).

3. Test data parameters

- Location information about the RemoteResourceGatekeeper
- Remote user's resource request
- Location information about the PolicyServer
- Policy request about the requested Grid resource
- Role based access policy/policies of the requested resource

5.5.1.1 TSF Tests

The TSF tests perform the testing on the functioning of the implemented TSF.Authorization as a whole, when called through the legitimate external interfaces.

1. Test Plan

The tests are on TSF.Authorization to see whether an authenticated user's request to access Grid resources can be handled correctly by the authorization function and whether the user will receive an appropriate response regarding the authorization as specified. The testing needs to consider all possible combinations of the different types of user requests, the related resource policy/policies, and the expected corresponding evaluation results.

2. Test Procedure Description

Different inputs to the external interfaces of TSF.Authorization – AuthorizationTSF::Configuration (location(s) of PolicyServer(s)) and AuthorizationTSF::acceptConnection (incoming connection from the RemoteClient, user request) shall be provided in the testing to check whether the evaluation results of TSF.Authorization match the expected results.

In the testing, the resource policy shown in Annex B is used to evaluate user requests for the policy's corresponding resource (but can also conduct tests on accessing a variety of other resources and performing various operations, which is not discussed here). This policy includes all essential elements for a basic XACML formatted role based resource access policy and considers environmental factors. The contents of these elements, especially those of the Rules, constitute a complete and effective set of contradiction-free constraints on the access to the resource, and all the effects of applying Rules will be combined using an appropriately defined combining algorithm in the evaluation to generate a final evaluation result.

When the RemoteResourceGatekeeper is started, the administrator needs to provide it with the location information about the relevant PolicyServer via the interface Configuration. Both correct and incorrect parameter values of the PolicyServer host address are input in repeated tests respectively. Then the remote user can connect with the RemoteResourceGatekeeper and send his resource request to TSF.Authorization via the interface acceptConnection by way of the RemoteClient. Valid and invalid user requests will also be provided respectively in repeated tests to see the final evaluation results returned. Thus, the total input to TSF.Authorization in each test is a combination of the correct/incorrect configuration data for the RemoteResourceGatekeepr and a valid/invalid user request. All possible combinations should be tested and the actual results will be matched with the expected results.

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8
IP Address	Correct	Correct	Correct	Correct	Incorrect	Incorrect	Incorrect	Incorrect
Port Number	Correct	Incorrect	Incorrect	Correct	Correct	Incorrect	Incorrect	Correct
User Request	Valid	Valid	Invalid	Invalid	Valid	Valid	Invalid	Invalid

Table 5.4 Test Cases for TSF.Authorization

For each input combination in the testing, it has three entries - IP address of the PolicyServer host, port number of the PolicyServer host, and the remote user's resource request. Eight possible combinations for the total input are shown in Table 5.4, and the eight Cases are tested for TSF.Authorization.

However, the invalid user request mentioned here and in this section only means a file incompatible with the XACML grammar, syntax or specification, such as a request lacking an essential element which is required by the XACML. As for the set of valid XACML formatted user requests, it includes both the requests that will be accepted and the requests that will be rejected by TSF.Authorization against the resource policy. So different types of valid user requests need to be tested within Cases 1, 2, 5, and 6, in order to determine whether TSF.Authorization only accepts a valid user request that satisfies all constraints stated in the resource policy and will reject any other valid user request even though it may only fail to satisfy one constraint from the resource policy.

Two tables are used to record the test results: one for the tests of Case 1, and the other for the tests of Cases 2 – 8.

3. Test Results

In Case 1, the decision of TSF.Authorization is made based on the contents of the user request and the resource policy, and the valid format of the user request is only the precondition for the request to be effectively interpreted and evaluated. Different user requests may get different evaluation results against the same resource policy because of their differences in the content. Since Subject, Resource, and Action are the three main content elements of a XACML formatted user request, different user requests with different contents in these elements are tested to see the evaluation results of

TSF.Authorization. The content of an element of the user request can be acceptable or unacceptable - acceptable content means that this element contains all essential attributes and attribute values recognized and required by the resource policy, and unacceptable content means that this element contains some attribute values which cannot be handled against the resource policy or lacks some essential attributes. In the testing, all possible combinations of the three elements' contents for a user request as well as the satisfaction of the possible Conditions stated in the resource policy should be considered. Such Sub-cases of Case 1 are listed in Table 5.5 and the testing results of them are shown in Table 5.6.

Element in the User Request	Subject	Resource	Action
Sub-case 1.1.1	Acceptable ("Zhu Ning, NTU postgraduate student")	Acceptable ("record of PDCC students")	Acceptable ("read")
	Received between 9:00 and 17:00 Singapore Time within a day		
Sub-case 1.1.2	Acceptable ("Zhu Ning, NTU postgraduate student")	Acceptable	Acceptable ("read")
	Received before 9:00 or after 17:00 Singapore Time within a day		
Sub-case 1.1.3	Acceptable ("Zhu Ning, NTU postgraduate student")	Acceptable	Acceptable ("delete" or "write")
Sub-case 1.1.4	Acceptable ("NTU staff")	Acceptable	Acceptable ("read")
Sub-case 1.1.5	Acceptable ("NTU staff")	Acceptable	Acceptable ("delete" or "write")
Sub-case 1.1.6	Acceptable ("XYZ, NTU postgraduate student")	Acceptable	Acceptable ("read")
Sub-case 1.1.7	Acceptable ("XYZ, NTU postgraduate student")	Acceptable	Acceptable ("delete" or "write")
Sub-case 1.2	Acceptable ("NTU postgraduate student" or "NTU staff")	Unacceptable (other resources, such as "record of PDCC staff")	Acceptable ("read" or "delete" or "write")
Sub-case 1.3	Acceptable ("NTU postgraduate student" or "NTU staff")	Unacceptable	Unacceptable (other actions, such as "execute")
Sub-case 1.4	Acceptable ("NTU postgraduate student" or "NTU staff")	Acceptable	Unacceptable
Sub-case 1.5	Unacceptable (having other values for the "role" attribute, such as "NTU alumni"; or even lacking the "role" attribute)	Acceptable	Acceptable ("read" or "delete" or "write")
Sub-case 1.6	Unacceptable	Unacceptable	Acceptable ("read" or "delete" or "write")
Sub-case 1.7	Unacceptable	Unacceptable	Unacceptable
Sub-case 1.8	Unacceptable	Acceptable	Unacceptable

Table 5.5 Sub-Cases of Case 1

	Actual Evaluation Results	Expected Evaluation Results
Sub-case 1.1.1 tests	Permit	Permit
Sub-case 1.1.2 tests	Not Applicable	Not Applicable
Sub-case 1.1.3 tests	Deny	Deny
Sub-case 1.1.4 tests	Permit	Permit
Sub-case 1.1.5 tests	Permit	Permit
Sub-case 1.1.6 tests	Not Applicable	Not Applicable
Sub-case 1.1.7 tests	Deny	Deny
Sub-case 1.2 tests	Not Applicable	Not Applicable
Sub-case 1.3 tests	Not Applicable	Not Applicable
Sub-case 1.4 tests	Not Applicable	Not Applicable
Sub-case 1.5 tests	Not Applicable	Not Applicable
Sub-case 1.6 tests	Not Applicable	Not Applicable
Sub-case 1.7 tests	Not Applicable	Not Applicable
Sub-case 1.8 tests	Not Applicable	Not Applicable

Table 5.6 Record of Testing Case 1

In the testing of Case 1, multiple relevant tests have been performed for each Sub-case because a Sub-case of Case 1 only represents one type of valid user request and may cover many different practical user requests of the same type. For example, the testing of Sub-case 1.1.3 includes both the tests where the user requests ask for a "delete" action and the tests where the user requests ask for a "write" action. In the testing of Sub-case 1.1.7, an involved user request may be sent by any "NTU postgraduate student" other than "Zhu Ning" and may ask for a "delete" action or a "write" action on the target resource, so the various user requests with same "role" but different "subject-id" or "action-id" attribute values are tested for the same Sub-case. For Sub-case 1.3, the user requests include those with different acceptable "role"s and different unacceptable "action-id"s for different unacceptable resources. In Sub-case 1.8, all different user requests with different unacceptable Subject or Action elements for the correct target resource are included.

For similar reasons, repeated relevant tests with different practical total inputs of the same type or other related information have also been performed for each of the other test Sub-cases and Cases in section 5.5.

Besides, the testing for Case 1 and the other test Cases in this section also considers the conditions about the environmental factors as prescribed by the resource policy. Sub-cases 1.1.1 and 1.1.2 are tested with the same type of user requests but in different time ranges. Their different testing results show that the actual effects of applying these conditions in the evaluation are as expected. For the other Sub-cases and Cases, the different times of receiving user requests do not make a difference in the final evaluation results of these requests, so such Sub-cases and Cases just include the associated tests performed in any possible time ranges.

	IP Address	Port Number	User Request
Sub-case 2.1	Correct	Incorrect	Valid (acceptable)
Sub-case 2.2	Correct	Incorrect	Valid (unacceptable Subject or Action element)
Sub-case 2.3	Correct	Incorrect	Valid (unacceptable Resource element)
Sub-case 5.1	Incorrect	Correct	Valid (acceptable)
Sub-case 5.2	Incorrect	Correct	Valid (unacceptable Subject or Action element)
Sub-case 5.3	Incorrect	Correct	Valid (unacceptable Resource element)
Sub-case 6.1	Incorrect	Incorrect	Valid (acceptable)
Sub-case 6.2	Incorrect	Incorrect	Valid (unacceptable Subject or Action element)
Sub-case 6.3	Incorrect	Incorrect	Valid (unacceptable Resource element)

Table 5.7 Sub-cases of Cases 2, 5, and 6

For Cases 2 – 8, the address provided for the PolicyServer host is incorrect or the input user request is not valid, so the resource policy cannot be retrieved or the user request cannot be interpreted for the evaluation. Then a “Not Applicable” or “Indeterminate” result will be returned to the RemoteClient and remote requesting user in the testing. Exceptions will also appear unless there is a valid user request with an unacceptable

Resource element. Such situations are demonstrated as corresponding Sub-cases in Table 5.7, and Table 5.8 shows the results of testing Cases 2 - 8.

	Actual Evaluation Results and Exceptions Caught	Expected Evaluation Results and Possible Exceptions
Sub-case 2.1 tests	Not Applicable, and ConnectException, NullPointerException, FileNotFoundException	Not Applicable, and ConnectException, NullPointerException, FileNotFoundException
Sub-case 2.2 tests	Not Applicable, and ConnectException, NullPointerException, FileNotFoundException	Not Applicable, and ConnectException, NullPointerException, FileNotFoundException
Sub-case 2.3 tests	Not Applicable	Not Applicable
Case 3 tests	Indeterminate (status: syntax-error), and ParsingException	Indeterminate, and ParsingException
Case 4 tests	Indeterminate (status: syntax-error), and ParsingException	Indeterminate, and ParsingException
Sub-case 5.1 tests	Not Applicable, and ConnectException, NullPointerException, FileNotFoundException	Not Applicable, and ConnectException, NullPointerException, FileNotFoundException
Sub-case 5.2 tests	Not Applicable, and ConnectException, NullPointerException, FileNotFoundException	Not Applicable, and ConnectException, NullPointerException, FileNotFoundException
Sub-case 5.3 tests	Not Applicable	Not Applicable
Sub-case 6.1 tests	Not Applicable, and ConnectException, NullPointerException, FileNotFoundException	Not Applicable, and ConnectException, NullPointerException, FileNotFoundException
Sub-case 6.2 tests	Not Applicable, and ConnectException, NullPointerException, FileNotFoundException	Not Applicable, and ConnectException, NullPointerException, FileNotFoundException
Sub-case 6.3 tests	Not Applicable	Not Applicable
Case 7 tests	Indeterminate (status: syntax-error), and ParsingException	Indeterminate, and ParsingException
Case 8 tests	Indeterminate (status: syntax-error), and ParsingException	Indeterminate, and ParsingException

Table 5.8 Record of Testing Cases 2 – 8

5.5.1.2 Testing of Subsystems

To achieve EAL 3, tests are also performed to show that the internal interfaces between subsystems of TSF.Authorization have been exercised and the subsystems work correctly according to TSF.Authorization's high-level design.

1. Test Plan

The testing is to find whether the input to an interface of a subsystem of TSF.Authorization will be handled correctly and lead to the expected effects and output.

2. Test Procedure Description

All interfaces of the six subsystems need to be tested. It should be checked that an input to the interface gets an appropriate output as specified in the high-level design. Most test cases for the testing of subsystems are already included in the test cases of TSF tests and for scenarios that are not covered, additional test cases are defined to test the relevant interfaces between the subsystems of TSF.Authorization. The testing for the internal interfaces can be addressed by testing via the external interfaces of TSF.Authorization. This is because in all subsystem tests, the possible input to each subsystem interface of TSF.Authorization is affected and determined by the input to the external interfaces.

3. Test Results

The testing of the Gatekeeper subsystem and the Gatekeeper_Communication subsystem are covered by TSF tests, but when considering the testing of the GetPolicy subsystem interfaces, it is found that the resource policy adopted for testing the authorization is assumed always available on the PolicyServer host for retrieval in the TSF tests (Cases 1 – 8), so that the TSF tests only address the test cases where the requested resource policy exists. As for situations where the requested resource policy does not exist, two new test cases, Cases 9 (three Sub-cases) and 10, are needed. In these two Cases, the input address of the related PolicyServer host should be correct because if not, it is meaningless to consider whether the requested resource policy exists on the PolicyServer host (Actually, Cases 2, 3, and 5 – 8 cover all possible test scenarios where an incorrect PolicyServer host address was input, addressing both the situations where the resource policy exists and the situations where the resource policy does not exist). Consequently, Cases 9 and 10 are similar to Case 1 and 4 except that in Cases 9 and 10, the requested resource policy cannot be found from the PolicyServer. The two test Cases are listed in Table 5.9 and their testing results are shown in Table 5.10.

	IP Address	Port Number	User Request	Requested Resource Policy
Sub-case 9.1	Correct	Correct	Valid (acceptable)	Not Exist
Sub-case 9.2	Correct	Correct	Valid (unacceptable Subject or Action element)	Not Exist
Sub-Case 9.3	Correct	Correct	Valid (unacceptable Resource element)	Not Exist
Case 10	Correct	Correct	Invalid	Not Exist

Table 5.9 Test Cases 9 and 10

	Actual Evaluation Results and Exceptions Caught	Expected Evaluation Results and Possible Exceptions
Sub-case 9.1 tests	Not Applicable, and NullPointerException, FileNotFoundException	Not Applicable, and NullPointerException, FileNotFoundException
Sub-case 9.2 tests	Not Applicable, and NullPointerException, FileNotFoundException	Not Applicable, and NullPointerException, FileNotFoundException
Sub-case 9.3 tests	Not Applicable	Not Applicable
Case 10 tests	Indeterminate (status: syntax-error), and ParsingException	Indeterminate, and ParsingException

Table 5.10 Record of Testing Cases 9 and 10

For the testing of the interfaces of the Evaluation subsystem, Cases 1 – 8 address situations where the acceptable resource policy exists to be used in the evaluation, and Cases 9 and 10 cover situations where the requested resource policy does not exist. However, for these internal interfaces, it is also possible that there only exists an invalid resource policy or valid resource policy with unacceptable content. The validity of a resource policy, similar to that of a user's resource request, only refers to the correctness of its XACML file format. As for whether the content of a retrieved valid resource policy is acceptable, it depends on the applicability of this policy to the resource requested by the user. For the testing of the new situations, four further test cases, Cases 11 – 14, are defined. Like Cases 9 and 10, Cases 11 – 14 are applicable only when the input PolicyServer host address is correct. These four Cases can also be regarded and tested as an enlargement to the set of the GetPolicy subsystem interfaces' test scenarios. Table 5.11 presents the four Cases in more detail and Table 5.12 records the testing results of them.

	IP Address	Port Number	User Request	Requested Resource Policy
Sub-case 11.1	Correct	Correct	Valid (acceptable)	Invalid
Sub-case 11.2	Correct	Correct	Valid (unacceptable Subject or Action element)	Invalid
Sub-case 11.3	Correct	Correct	Valid (unacceptable Resource element)	Invalid
Sub-case 12.1	Correct	Correct	Valid (acceptable)	Valid but unacceptable
Sub-case 12.2	Correct	Correct	Valid (unacceptable Subject or Action element)	Valid but unacceptable
Sub-case 12.3	Correct	Correct	Valid (unacceptable Resource element)	Valid but unacceptable
Case 13	Correct	Correct	Invalid	Invalid
Case 14	Correct	Correct	Invalid	Valid but unacceptable

Table 5.11 Test Cases 11 – 14

	Actual Evaluation Results and Exceptions Caught	Expected Evaluation Results and Possible Exceptions
Sub-case 11.1 tests	Indeterminate (status: syntax-error), and ParsingException	Indeterminate, and ParsingException
Sub-case 11.2 tests	Indeterminate (status: syntax-error), and ParsingException	Indeterminate, and ParsingException
Sub-case 11.3 tests	Not Applicable	Not Applicable
Sub-case 12.1 tests	Not Applicable, and ProcessingException	Not Applicable, and ProcessingException
Sub-case 12.2 tests	Not Applicable, and ProcessingException	Not Applicable, and ProcessingException
Sub-case 12.3 tests	Not Applicable	Not Applicable
Case 13 tests	Indeterminate (status: syntax-error), and ParsingException	Indeterminate, and ParsingException
Case 14 tests	Indeterminate (status: syntax-error), and ParsingException	Indeterminate, and ParsingException

Table 5.12 Record of Testing Cases 11 - 14

Finally, the testing of the interfaces of the PolicyGatekeeper subsystem and the PolicyFinder subsystem also needs to consider both the TSF test Cases and the relevant new test Cases, in order to cover all possible test scenarios faced by the interfaces and the subsystems.

In brief, the testing of the interfaces of TSF subsystems is addressed by the TSF test Cases and new Cases 9 - 14, as shown in Table 5.13. In the table, an invalid policy request is the one that cannot be interpreted by the PolicyServer, and a correct policy request is valid and asking for the access policy of the resource requested by a remote user. The new Cases 9 – 14 can also be used to enrich the testing of the Gatekeeper and Gatekeeper_Communication Subsystems.

Subsystem::Interface	Test Cases	Possible Input to the Interface	Possible Output of the Interface
Gatekeeper:: Configuration	1 – 8	location(s) of PolicyServer(s) (correct or incorrect)	
Gatekeeper::accept IncomingConnection	1 – 14	incoming connection from the RemoteClient	
Gatekeeper::Internal Communication	1 – 14	user request (acceptable or unacceptable or invalid)	evaluation result
Gatekeeper_ Communication::External Communication	1 – 14	user request (acceptable or unacceptable or invalid)	evaluation result
GetPolicy::retrievePolicy	1 – 14	policy request (correct or incorrect or invalid), location of PolicyServer (correct or incorrect)	
Evaluation::Eva	1 – 14	user request (acceptable or unacceptable)	evaluation result
Evaluation::EvaPolicy	1, 4, 9 – 14	resource policy/policies (acceptable or unacceptable or invalid or not exist)	
PolicyGatekeeper::accept PolicyRequest	1, 4, 9 – 14	incoming connection from the RemoteResourceGatekeeper, policy request (correct or incorrect or invalid)	found resource policy/policies or a “policy not found” response
PolicyFinder::handle PolicyRequest	1, 4, 9 – 14	policy request (correct or incorrect or invalid)	found resource policy/policies or a “policy not found” response

Table 5.13 Test Cases for the Subsystem Interfaces

In the testing, some code have also been added to the subsystem interfaces' implementations to print out the actual input and output data of each interface for comparison with the expected ones in all possible cases, as a more detailed verification of the proper working of these interfaces and subsystems conforming to the high-level design of TSF.Authorization. Full information, data and results of such verifications are not presented here for the reason of thesis length.

5.5.2 Analysis of the Test Coverage

Along with the functional testing for TSF.Authorization, the completeness of test coverage should also be addressed. It refers to the extent to which the TOE security function is tested and whether or not the testing is sufficiently extensive to demonstrate that the TOE security function operates as specified.

The TSF.Authorization has been tested against its functional specification in a systematic manner, to guarantee EAL 3 in its implementation. The TOE security function was repeatedly tested with fourteen test cases to demonstrate that it works fully as described in the functional specification. As recorded in subsection 5.5.1, these test cases include all possible combinations of the inputs to the external interfaces of TSF.Authorization with the possible environmental factors, to determine whether the corresponding effects and outputs of TSF.Authorization are correct and as derived from the functional specification. Each test case is essential for TSF.Authorization and covers a different and exclusive test scenario, so none of the test cases can be absent in the testing.

5.5.3 Analysis of the Depth of Testing

The level of detail to which the TSF is tested is another issue for test assurance. It is to counter the risk of missing an error in the development and help discover any possible malicious code that has been inserted. From the tests that exercise the specific internal interfaces (i.e. the interfaces of the subsystems), assurance can be provided that not only the TSF exhibits its desired external security behavior, but also that this behavior stems from correctly operating internal mechanisms.

Consequently, as the requirements of EAL 3, the testing performed also covers tests at the level of the subsystems of TSF.Authorization. The subsystems provide a high-level description of the internal workings of the TSF, and the testing on them will demonstrate the presence of possible flaws and bring assurance that the TSF subsystems have been correctly realized.

From the test results shown in subsection 5.5.1, it can be seen that the implementation of the TSF subsystems is consistent with the high-level design and that the internal interfaces of the subsystems operate correctly. The high-level design describes each TSF subsystem and all interfaces between the subsystems in detail and all internal interfaces have been exercised properly in the testing of Cases 1 - 14. The testing was conducted via the external interfaces of TSF.Authorization, considering all combinations of the possible inputs to the interfaces of the subsystems and comparing the actual TSF external outputs with the expected ones. Additional code was also added to the implementation package to print out the actual inputs and outputs of each internal interface for visual checking of TSF.Authorization's internal data transmission and operation in the testing.

Chapter 6

Conclusions and Future Work

6.1 Summary

The Grid is a fast evolving novel form of distributed computing. It allows a great number of distributed heterogeneous computer systems to work together towards large scale problem solving and resource sharing. Currently, Grid computing is developing to encompass a new set of abilities such as B2B and B2C transactions. However, although security problems are becoming increasingly important, existing Grid security solutions do not address security in a systematically assured way which is required for trustworthy implementations. It is also easy to make mistakes and create vulnerable business and industry oriented Grid systems/applications because of the simple security functionality originally designed for academic Grids. The trustworthy secure Grids need a disciplined security engineering development process to preserve their specific security objectives.

In this thesis, basic concepts of Grid computing have been reviewed and existing typical Grid security solutions surveyed. It was found that in addition to their functional inadequacies, these security solutions were developed proprietarily without a disciplined process so that they cannot assure the correct satisfaction of the security requirements and specific security objectives. Current Grid security implementations are usually application specific without consistent and systematic standards, methods and modeling for high level analysis of the security requirements and risks in their development and evaluation. This implies that their scope of acceptance will be narrow and it is hard to ensure the required trustworthiness. In other words, the developers can hardly prove the extent and strength level of the security that their products provide and whether the product is appropriate and meets all their claimed security objectives. It is also not easy for developers to reuse or learn from the products of other people for a new Grid security implementation. Consequently, a generic Grid security assurance framework is proposed for the development of trustworthy Grids.

To set up the security assurance framework, ISO/IEC international standard 15408, the Common Criteria, was followed in the disciplined security specifications and design for the Grid. A security environment was conceptualized to provide the stimulation to establish the Grid security objectives, and security functional and assurance requirements were further derived, refined, and grouped systematically for the Grid. They were all presented with the appropriate rationale for justification. These generic security specifications within the security assurance framework form the foundation for the trustworthy Grid security development.

In a specific development process following this CC validated framework, a TOE summary specification is defined based on the disciplined Grid security functional and assurance requirements. Further instantiations from the TOE summary specification to a less abstract TOE representation is carried out step by step, from the TOE security functions summary to the functional specification, to the high-level design, to the low-level design, and finally to the representation of the implementation. This process will lead to a trustworthy Grid TOE security implementation which is assured to meet the security objectives by enforcing the defined assurance measures in the development. Such a specific security implementation can utilize more powerful security techniques and deploy application or scenario specific configurations, to provide not only high assurance but also improved functionality for the Grid systems/applications.

The development of a sample TOE security function following the Grid security assurance framework, TSF.Authorization for the role based authorization at EAL 3, has been demonstrated in this thesis. From the functional viewpoint, the TOE security function implements flexible fine grained access control to provide better protection of Grid resources than most academic Grid security architectures and implementations. From the assurance viewpoint, all relevant assurance measures to implement EAL 3 have been properly enforced. The assurance evidence demonstrates that the development process is systematic, and all the design specifications were disciplined according to the relevant security assurance requirements and were justified as consistent and suitable. Life cycle security relevant considerations were also addressed and life cycle support assurance was achieved in the development. The complete tests have helped us find errors and mistakes in the implementation, and verified that the TOE security function and its subsystems work well as designed in the abstract TOE representations. This sample TOE security function is a trustworthy one based on EAL 3.

In the specific development process, a full TOE summary specification has been given for the whole Grid TOE security implementation, but only a systematic instantiation and implementation for TSF.Authorization with stated assurance has been presented. The trustworthy development of other defined security functions for this Grid TOE can be performed in a similar disciplined way as that for TSF.Authorization, to construct a complete Grid security implementation at EAL 3 following the proposed Grid security assurance framework. The generic Grid security specifications were systematically derived from a thorough security analysis of a number of typical computational Grids and presented in a way that conforms to the international recognized IT security evaluation standard and assurance method, the Common Criteria. The whole Grid security assurance framework for trustworthy development was also disciplined and validated with the Common Criteria. Consequently, this security assurance framework could be applied as a systematic basis and reference for the flexible development of a variety of trustworthy computational Grid systems, especially a generic Grid-based application execution and information sharing engine such as ClearingHouse. The resulting Grid security implementations may realize different specific security features and strength levels by adopting different security technical tools and models, but all of them will hold high assurance by correctly enforcing corresponding assurance measures in their security engineering development processes. Wide recognizing and easy reuse of them by communities and organizations can thus be achieved.

6.2 Advantages and Originality

Security is an issue covering multiple areas and can be considered at multiple levels. This thesis has addressed security assurance issues for the Grid and proposed a CC validated Grid security assurance framework for disciplined development of trustworthy Grid security implementations in fulfillment of a variety of security requirements. These are theoretical contributions of this research work. Other Grid security researchers have only considered functional issues and technical implementations and have not followed a systematic approach in their security specifications, designs, implementations, tests, deployments and operations. Grid security was thus only addressed at a lower level sometimes providing even proprietary security architectures and implementations without disciplined high level analysis of security functional needs and assurance considerations.

This may lead to an untrustworthy implementation and the claimed security functionality is not assured to be correctly fulfilled according to the design specifications and meeting the claimed security objectives.

In the practical implementation, TSF.Authorization fulfills a role based access control for more powerful and flexible Grid security, and the three components of the implementation package can coordinate well with the relevant hardware, software and middleware on their host machines. An XACML language is also adopted to express authorization relevant information for fine grained authorization based on flexibly distributed resource access policies. These cater to the security technical needs of the Grids evolving toward a larger scale global distributed environment and oriented to business and industry applications which require enriched security functionality. In contrast, current authorization mechanisms and security solutions for academic Grids do not address them well. The complete Grid security architectures and solutions with well enhanced and highly assured functionality can be designed, implemented and evaluated through our disciplined security engineering processes. All these are practical contributions of this research work.

All of the contributions distinguish the work presented in this thesis from and gain an advantage over the past and other existing Grid security efforts. My research aims to develop and evaluate Grid security at a high and abstract level, which has not been conducted by the Grid community. The disciplined and systematically derived generic Grid security specifications and assurance framework will guide and justify the development, and finally bring a trustworthy security solution with better functionality for the evolving Grid.

6.3 Limitations

The security assurance framework proposed in this thesis for trustworthy Grid security development still has some limitations. For example, the derivation of generic Grid security specifications is only based on the study and analysis of the existing Grid projects that have been surveyed, and the disciplined security functional requirements have only been grouped into three subsets related to a rough division of three basic strength levels of security functionality for flexible development of different Grid security solutions and scenarios. Existing justifications of the security specifications, assurance framework and disciplined development are only expressed in informal and semi-formal ways. Consequently, both surveys on more Grid systems/applications and experiences from more practical security development are needed. This will improve, enrich and elaborate the generic Grid security specifications and assurance framework with appropriate coverage and justification for providing more suitable and flexible trustworthy Grid security development. In addition, more justification approaches may need to be introduced.

Regarding the Common Criteria itself – the basis of the proposed Grid security assurance framework and the security engineering international standard being followed, it also has some limitations. For example, CC was initially designed mainly for single IT security products and secure operating systems. However, CC has already proven its further suitability and applicability to multiple domain solutions, mobile code, network management and security management [CCEVS] which are features of Grid scenarios. This provides a good starting point and background for applying CC to the Grid. The current version of the generic Grid security specifications and assurance framework primarily focuses on trustworthy security development and implementation at the Grid and resource domain level and of the Grid portals or security gatekeepers. Larger and deeper coverage of the highly assured Grid security may be needed in the future.

6.4 Future Work

In the future work, a complete and formal Protection Profile will be built for the generic Grid-based application execution and information sharing engines with a complete set of rationales based on the existing generic Grid security specifications. A variety of Security Targets following this PP and TOEs' specific security technical needs and summary specifications can thus be produced for the security development of different trustworthy Grid TOEs such as those described in [FGK04].

To address the limitations discussed in the previous section, it is necessary to study further various Grid systems and applications to derive clearer, more precise, and more concrete disciplined generic specifications for the Grid security environmental artifacts and security objectives, and to better and more specifically define, extend, tailor, and group Grid security functional and assurance requirements (such as utilizing the CLASP (Comprehensive, Lightweight Application Security Process) [CLASP] compliant approach proposed in [VIE05]). Practical experiences gained from the security development of trustworthy Grid systems/applications will also be input to the refining process of the Grid security specifications and assurance framework. The refining can be an iterative process that repeatedly absorbs both theoretical results (such as [BMM05] and [DGL05]) and practical needs, to be able to guide producing diverse trustworthy Grid security implementations with different EALs more effectively and flexibly. Moreover, in the future work the same security engineering approach can even be followed to construct new generic security specifications, Protection Profiles, Security Targets and assurance frameworks for the development of trustworthy Grid systems/applications of new scenarios or categories. Characterization of the development efforts, although very hard, may also be explored for the framework [SMI01]. If needed, a Grid specific extension of the Common Criteria may be produced to address more Grid security technical and assurance needs and issues at all levels. In addition, the process assurance methods and/or environmental assurance methods can be used together with the Common Criteria to construct a hybrid security engineering process and a more comprehensive assurance framework as some Grid aspects not covered by the current Grid security assurance framework can be addressed by other security engineering standards and methods. For instance, people can utilize environmental assurance methods introduced in subsection 2.2.2 and security management international standards (such as ISO/IEC 17799 [SM1] and BS 7799.2 [SM2]) to assure the trustworthy implementation of host level security for the Grid.

New modeling, specification, and justification strategies, methods, or tools may also be used or developed in the future, for more effective and efficient derivation of the security specifications and for trustworthy development following the Grid security assurance framework. For example, a diagram-based modeling method that uses UML use cases and UML security extension [UMLSEC] [JÜR02] [JÜR04] can be used to describe the disciplined generic security environment and objectives for the Grid. UML use case diagrams will model system functionality at a very high level of abstraction, and define system boundaries and external systems and actors that are active within the context. This approach may enable us to specify consistent security environment and objectives more clearly and precisely in an easily readable and understandable manner, and provide a convenient and systematic way to generate, refine and justify further security artifacts and dependencies for the TOE.

TSF.Authorization, the TOE security function developed and demonstrated for role based authorization, mainly addresses a relatively simple scenario where the user's role/attributes are incorporated in his resource request and the resource access policy/policies are retrieved from a single PolicyServer. The implementation can be extended to suit more complex scenarios. For example, the user's role/attributes can be signed by an attribute authority and stored on a remote AttributeServer, and the trusted access policies of a resource may be dispersed on several PolicyServer hosts. In such cases, a user's request may contain the information about the requested resource and action with or without his role/attributes. Upon receipt of a user request, the RemoteResourceGatekeeper can use an additional GetAttribute subsystem (similar to the GetPolicy subsystem) to contact the proper AttributeServer to get the trusted role/attributes information for the remote user, and invoke the GetPolicy subsystem to collect the complete set of the requested resource's access policies from all related PolicyServers.

As for the current Evaluation subsystem implementation, it already supports the input of more than one XACML formatted resource policy, and can be further extended to accept more complex inputs. Then the remote user's resource request and his trusted role/attributes will be evaluated against all retrieved resource policies, and the possible user role/attributes information contained in the user request should also be compared with that trusted one received from the AttributeServer. The evaluation results of the user request against each access policy will be combined following the policy combining algorithm to form the final result of evaluating the user request against a policy set.

Besides the possible technical extension to the TSF.Authorization implementation, the effective and efficient implementation of the other security functions defined by the TOE summary specification in section 5.1 for the specific Grid TOE also needs powerful and appropriate security engineering techniques and methods, to fulfill justified security designs and provide corresponding assurance evidence. Configuration management, delivery and operation, guidance documents, and vulnerability relevant assurance measures should be enforced for the complete trustworthy Grid TOE security implementation. All the Grid security functions must be implemented in a way to work cooperatively with each other and with the environment and underlying hardware/software/middleware infrastructures, to fully function and properly secure a Grid system (such as the ClearingHouse and Nanyang Campus Grid) and related applications.

In the future work, according to the flexible security needs of different Grid use cases and defined PPs/STs, it will be possible to conduct various disciplined development to produce different specific security solutions and implementations with different security strengths and EALs. For the evolving Grid, additional or enhanced implementations of those security functions which were less technically addressed by existing Grid security solutions can even be included to meet the corresponding security objectives and functional requirements.

Finally, to develop not only highly assured but also more flexible and powerful Grid security functionality, new security techniques, tools, and models may also be required. XACML has been chosen to express authorization relevant context and policy information for our Grid security implementation, and it is also found that the SAML (Security Assertion Markup Language) [SAML] language defines a XML-based message format and protocol and can be used to exchange authentication, attribute, and authorization information about principals in Grid security implementations. So an extended SAML or XACML language or a combined new XML-based information format may even be developed to better express and exchange security relevant information (e.g. identity, role/attributes, policy, etc.) and perform security relevant operations (e.g. authentication, authorization, etc.) for specific Grid security systems in the future. Some Grid simulation tools (such as G3S (Grid Security Services Simulator) [NR05]) can also be used in the Grid security development. Besides, it has been noticed that evolved security management models may be necessary for Grids in business settings because of the distinctive features of complex business environments, and resource trust level mechanisms may even be introduced to manage the admittance and expelling of the resources for the Grid. Cache and other relevant configurations and techniques (such as securing local resource hosts from malicious applications [DSG04] [ZBP05]) can also be used in the implementation of the Grid TSFs and TOEs to advance the efficiency and performance of the security operations, processes and mechanisms.

Annex A A Sample Resource Request in XACML Format

```
<?xml version="1.0" encoding="UTF-8" ?>

<Request xmlns="urn:oasis:names:tc:xacml:1.0:context"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context cs-
         xacml-schema-context-01.xsd">

  <Subject>
   <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
     DataType="urn:oasis:names:tc:xacml:1.0:data-type:x500Name"
     Issuer="Nanyang Technological University">
     <AttributeValue>O=Grid, OU=NTUCG, OU=NTUCG-ntuca.ntu.edu.sg,
     OU=ntu.edu.sg, CN=Zhu Ning</AttributeValue>
   </Attribute>
   <Attribute AttributeId="role"
     DataType="http://www.w3.org/2001/XMLSchema#string"
     Issuer="Nanyang Technological University">
     <AttributeValue>NTU postgraduate student</AttributeValue>
   </Attribute>
  </Subject>

  <Resource>
   <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-
     id"
    DataType="http://www.w3.org/2001/XMLSchema#anyURI">
    <AttributeValue>http://www.ntu.edu.sg/pdcc/students</AttributeValue>
   </Attribute>
  </Resource>

  <Action>
   <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
     DataType="http://www.w3.org/2001/XMLSchema#string">
     <AttributeValue>read</AttributeValue>
   </Attribute>
  </Action>

</Request>
```

Annex B A Sample Resource Policy in XACML Format

```
<?xml version="1.0" encoding="UTF-8" ?>

<Policy  xmlns="urn:oasis:names:tc:xacml:1.0:policy"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy cs-xacml-
         schema-policy-01.xsd"

        PolicyId="urn:oasis:names:tc:xacml:1.0:pdcc_students_record:policy"

        RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
         algorithm:deny-overrides">

 <Description>Policy for accessing the record of PDCC
   students.</Description>

 <Target>

     <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
           equal">
           <AttributeValue
             DataType="http://www.w3.org/2001/XMLSchema#string">NTU
             postgraduate student</AttributeValue>
           <SubjectAttributeDesignator
             AttributeId="role"
             DataType="http://www.w3.org/2001/XMLSchema#string" />
         </SubjectMatch>
      </Subject>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
           equal">
           <AttributeValue
             DataType="http://www.w3.org/2001/XMLSchema#string">NTU
             staff</AttributeValue>
           <SubjectAttributeDesignator
             AttributeId="role"
             DataType="http://www.w3.org/2001/XMLSchema#string" />
         </SubjectMatch>
      </Subject>
     </Subjects>

     <Resources>
      <AnyResource />
     </Resources>

     <Actions>
      <AnyAction />
     </Actions>
```

```
    </Target>

   <Rule RuleId="urn:oasis:names:tc:xacml:1.0:pdcc_students_record:rule1"
     Effect="Permit">
     <Description>Only the postgraduate student Zhu Ning of Nanyang
       Technological University can read the record of PDCC's students from
       9am to 5pm Singapore Time every day.</Description>

      <Target>

         <Subjects>
           <Subject>
             <SubjectMatch
                 MatchId="urn:oasis:names:tc:xacml:1.0:function:x500Name-
                 equal">
                 <AttributeValue
                   DataType="urn:oasis:names:tc:xacml:1.0:data-
                   type:x500Name">O=Grid, OU=NTUCG, OU=NTUCG-
                   ntuca.ntu.edu.sg, OU=ntu.edu.sg, CN=Zhu
                   Ning</AttributeValue>
                <SubjectAttributeDesignator
                   AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-
                   id" DataType="urn:oasis:names:tc:xacml:1.0:data-
                   type:x500Name" />
             </SubjectMatch>
             <SubjectMatch
                 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
                 equal">
                 <AttributeValue
                   DataType="http://www.w3.org/2001/XMLSchema#string">
                   NTU postgraduate student</AttributeValue>
                 <SubjectAttributeDesignator
                   AttributeId="role"
                   DataType="http://www.w3.org/2001/XMLSchema#string"
                   />
              </SubjectMatch>
            </Subject>
          </Subjects>

          <Resources>
            <Resource>
              <ResourceMatch
                 MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-
                 equal">
                 <AttributeValue
                   DataType="http://www.w3.org/2001/XMLSchema#anyURI"
                   >http://www.ntu.edu.sg/pdcc/students</AttributeValue>
                 <ResourceAttributeDesignator
                   AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource
                   -id"
                   DataType="http://www.w3.org/2001/XMLSchema#anyURI"
                   />
              </ResourceMatch>
             </Resource>
           </Resources>

           <Actions>
             <Action>
            <ActionMatch
              MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
              equal">
              <AttributeValue
                DataType="http://www.w3.org/2001/XMLSchema#string">
                read</AttributeValue>
              <ActionAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                DataType="http://www.w3.org/2001/XMLSchema#string"
                />
            </ActionMatch>
          </Action>
        </Actions>

      </Target>

      <Condition
        FunctionId="http://research.sun.com/projects/xacml/names/functi
        on#time-in-range">
         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-
          and-only">
           <EnvironmentAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-
            time"
            DataType="http://www.w3.org/2001/XMLSchema#time" />
         </Apply>
         <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#time">09:00:
            00</AttributeValue>
         <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#time">17:00:
            00</AttributeValue>
      </Condition>

    </Rule>

    <Rule RuleId="urn:oasis:names:tc:xacml:1.0:pdcc_students_record:rule2"
      Effect="Deny">
      <Description>Postgraduate students of Nanyang Technological University
        cannot delete or write the record of PDCC's students.</Description>

      <Target>

        <Subjects>
          <Subject>
            <SubjectMatch
                MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
                equal">
                <AttributeValue
                  DataType="http://www.w3.org/2001/XMLSchema#string">
                  NTU postgraduate student</AttributeValue>
                <SubjectAttributeDesignator
                  AttributeId="role"
                  DataType="http://www.w3.org/2001/XMLSchema#string"
                  />
            </SubjectMatch>
          </Subject>
        </Subjects>
      <Resources>
        <Resource>
          <ResourceMatch
              MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-
              equal">
              <AttributeValue
                DataType="http://www.w3.org/2001/XMLSchema#anyURI"
                >http://www.ntu.edu.sg/pdcc/students</AttributeValue>
              <ResourceAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource
                -id"
                DataType="http://www.w3.org/2001/XMLSchema#anyURI"
                />
          </ResourceMatch>
        </Resource>
      </Resources>

      <Actions>
        <Action>
          <ActionMatch
              MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
              equal">
              <AttributeValue
                DataType="http://www.w3.org/2001/XMLSchema#string">
                delete</AttributeValue>
              <ActionAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                DataType="http://www.w3.org/2001/XMLSchema#string"
                />
          </ActionMatch>
        </Action>
        <Action>
          <ActionMatch
              MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
              equal">
              <AttributeValue
                DataType="http://www.w3.org/2001/XMLSchema#string">
                write</AttributeValue>
              <ActionAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                DataType="http://www.w3.org/2001/XMLSchema#string"
                />
          </ActionMatch>
        </Action>
      </Actions>

    </Target>

  </Rule>

  <Rule RuleId="urn:oasis:names:tc:xacml:1.0:pdcc_students_record:rule3"
    Effect="Permit">
    <Description>Staff of Nanyang Technological University can read, delete
      and write the record of PDCC's students.</Description>

    <Target>

      <Subjects>
      <Subject>
        <SubjectMatch
              MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
              equal">
              <AttributeValue
                DataType="http://www.w3.org/2001/XMLSchema#string">
                NTU staff</AttributeValue>
              <SubjectAttributeDesignator
               AttributeId="role"
                DataType="http://www.w3.org/2001/XMLSchema#string"
                />
          </SubjectMatch>
        </Subject>
     </Subjects>

     <Resources>
        <Resource>
          <ResourceMatch
              MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-
              equal">
              <AttributeValue
                DataType="http://www.w3.org/2001/XMLSchema#anyURI"
                >http://www.ntu.edu.sg/pdcc/students</AttributeValue>
              <ResourceAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource
                -id"
                DataType="http://www.w3.org/2001/XMLSchema#anyURI"
                />
           </ResourceMatch>
         </Resource>
       </Resources>

       <Actions>
         <Action>
           <ActionMatch
              MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
              equal">
              <AttributeValue
                DataType="http://www.w3.org/2001/XMLSchema#string">
                read</AttributeValue>
              <ActionAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                DataType="http://www.w3.org/2001/XMLSchema#string"
                />
           </ActionMatch>
         </Action>
        <Action>
           <ActionMatch
              MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
              equal">
              <AttributeValue
                DataType="http://www.w3.org/2001/XMLSchema#string">
                delete</AttributeValue>
              <ActionAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                DataType="http://www.w3.org/2001/XMLSchema#string"
                />
           </ActionMatch>
```

```
            </Action>
            <Action>
              <ActionMatch
                 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
                 equal">
                 <AttributeValue
                   DataType="http://www.w3.org/2001/XMLSchema#string">
                   write</AttributeValue>
                 <ActionAttributeDesignator
                   AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                   DataType="http://www.w3.org/2001/XMLSchema#string"
                   />
              </ActionMatch>
            </Action>
          </Actions>

        </Target>

    </Rule>

</Policy>
```

Annex C A Sample Evaluation Result of Authorization in XACML Format

```
<?xml version="1.0" encoding="UTF-8" ?>

<Response  xmlns="urn:oasis:names:tc:xacml:1.0:context"
           xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context cs-
           xacml-schema-policy-01.xsd"

  <Result ResourceID="http://www.ntu.edu.sg/pdcc/students">
   <Decision>Permit</Decision>
  <Status>
    <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok" />
   </Status>
 </Result>

</Response>
```

Bibliography

[ACC03] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, Á. Frohner, A. Gianoli, K. Lõrentey, F. Spataro. VOMS, an Authorization System for Virtual Organizations. *1st European Across Grids Conference*, pages 33 – 40, 2003.

[ACGRID] The Access Grid project, http://www.accessgrid.org.

[ADF01] G. Allen, T. Dramlitsch, I. Foster, N. Karonis, M. Ripeanu, E. Seidel, B. Toonen. Supporting Efficient Execution in Heterogeneous Distributed Computing Environments with Cactus and Globus. *Supercomputing Conference 2001*, pages 52 - 76, 2001.

[AGK00] D. Abramson, J. Giddy, L. Kotler. High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid? *IEEE International Parallel and Distributed Processing Symposium 2000*, pages 520 – 528, 2000.

[AND01] R. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems.* John Wiley & Sons, 2001.

[ASSUR1] *Information Technology - Security Techniques - A Framework for IT Security Assurance - Part 1: Overview and Framework.* ISO/IEC 15443-1.

[ASSUR3] *Information Technology - Security Techniques - A Framework for IT Security Assurance - Part 3: Analysis of Assurance Methods.* ISO/IEC

15443-3.

[BAG00] R. Buyya, D. Abramson, J. Giddy. Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid. *International Conference on High Performance Computing in Asia Pacific Region 2000*, pages 283 – 289, 2000.

[BBL02] M. Baker, R. Buyya, D. Laforenza. Grid and Grid Technologies for Wide-Area Distributed Computing. *Software - Practice and Experience*, 32 (15), pages 1437 – 1466, 2002.

[BGA00] R. Buyya, J. Giddy, D. Abramson. An Evaluation of Economy-based Resource Trading and Scheduling on Computational Power Grids for Parameter Sweep Applications. *2nd Workshop on Active Middleware Services*, pages 41 – 50, 2000.

[BMM05] S.Banerjee, C. Mattmann, N. Medvidovic, L. Golubchik. Leveraging Architectural Models to Inject Trust into Software Systems. *ACM SIGSOFT Software Engineering Notes*, 30(4), pages 8 – 14, 2005.

[BWE00] R. Butler, V. Welch, D. Engert, I. Foster, S. Tuecke, J. Volmer, C. Kesselman. A National-Scale Authentication Infrastructure. *IEEE Computer*, 33(12), pages 60 - 66, 2000.

[CPGRID] Nanyang Campus Grid project, http://ntu-cg.ntu.edu.sg.

[CAPP] *Controlled Access Protection Profile, Version 1.d.* National Security Agency, Information Systems Security Organization, http://niap.nist.gov/ccscheme/PP_CAPP_V1.d.pdf.

[CC1] *Information Technology-Security Techniques - Evaluation Criteria for IT Security - Part 1: Introduction and General Model.* ISO/IEC 15408-1.

[CC2] *Information Technology - Security Techniques-Evaluation Criteria for IT Security - Part 2: Security Functional Requirements.* ISO/IEC 15408-2.

[CC3] *Information Technology-Security Techniques - Evaluation Criteria for IT Security - Part 3: Security Assurance Requirements.* ISO/IEC 15408-3.

[CCEVS] *The NIAP Common Criteria Evaluation and Validation Scheme for IT Security Program.* http://niap.nist.gov/cc-scheme/index.html.

[CCRA] *Arrangement on the Recognition of Common Criteria Certificates in the Field of Information Technology Security.* http://niap.nist.gov/cc-scheme/ccra.pdf.

[CEM1] *Common Evaluation Methodology for Information Technology Security CEM - 97/017 Part 1: Introduction and General model, Version 0.6.*

[CEM2] *Common Evaluation Methodology for Information Technology Security CEM - 99/045 Part 2: Evaluation Methodology, Version 1.0.*

[CLASP] CLASP (Comprehensive, Lightweight Application Security Process), http://www.securesoftware.com/CLASP/.

[CLHOUSE] The ClearingHouse project, http://ntu-cg.ntu.edu.sg/ClearingHouse/index.html.

[CMM] *Capability Maturity Model for Software.* CMU/SEI-91-TR-24, Software Engineering Institute, Carnegie Mellon University, 1991.

[CO02] D. Chadwick, O. Otenko. The PERMIS X.509 Role Based Privilege Management Infrastructure. *ACM Symposium on Access Control Models and Technologies 2002*, pages 135 – 140, 2002.

[COB03] D. Chadwick, A. Otenko, E. Ball. Role-based Access Control with X.509 Attribute Certificates. *IEEE Internet Computing 7(2)*, pages 62 – 69, 2003.

[DEGPSE] The DEGPSE project, http://xeon.ntu.edu.sg/degpse.html.

[DG] The DataGrid project, http://www.eu-datagrid.org.

[DGL05] Y. Demchenko, L. Gommans, C. Laat, B. Oudenaarde. Web Services and Grid Security Vulnerabilities and Threats Analysis and Model. *IEEE/ACM International Workshop on Grid Computing 2005*, pages 262 – 267, 2005.

[DSG04] E. Dodonov, J. Sousa, H. Guardia. GridBox: Securing Hosts from Malicious and Greedy Applications. *2nd Workshop on Middleware for Grid Computing*, pages 17 – 22, 2004.

[FGK04] I.Foster, D.Gannon, H.Kishimoto, J.Reich. Open Grid Services Architecture Use Cases. GGF Specification, 2004.

[FIPS] *Security Requirements for Cryptographic Modules.* Federal Information Processing Standards Publication FIPS 140-2.

[FK98] I. Foster, C. Kesselman. *The Grid: Blueprint for a Future Computing Infrastructure.* Morgan Kaufman, 1998.

[FKN02] I. Foster, C. Kesselman, J. Nick, S. Tuecke. The Physiology of the Grid: Open Grid Services Architecture for Distributed Systems Integration. *4th Global Grid Forum*, 2002.

[FKT98] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke. A Security Architecture for Computational Grids. *5th ACM Conference on Computer and Communication Security*, pages 83 – 92, 1998.

[FKT01] I. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid. *Intentional Journal of Supercomputer Applications*, 15(3), pages 200 – 222, 2001.

[FOS05] I.Foster. Globus Tookit Version 4: Software for Service-Oriented Systems. *IFIP International Conference on Network and Parallel Computing 2005,* pages 2-13, 2005.

[FSH03] I. Flechais, M. Sasse, S. Hailes. Bringing Security Home: A Process for Developing Secure and Usable Systems. *ACM Workshop on New Security Paradigms 2003*, pages 49 – 57, 2003.

[GCG02] D. Gannon, K. Chiu, M. Govindaraju, A. Slominski. A Revised Analysis of the Open Grid Services Infrastructure. http://www.extreme.indiana.edu/~gannon/ogsaAnalysis4.pdf, 2002.

[GEL04] M. Geldof. The Semantic Grid: Will Semantic Web and Grid Go Hand in Hand? June 2004.

[GR02] C. Goble, D. Roure. The Grid: An Application of the Semantic Web. *SIGMOD Record*, 31(4), 2002.

[GLOBUS] The Globus project, http://www.globus.org.

[GRI97] A. Grimshaw et al. The Legion Vision of a Worldwide Virtual Computer. *Communications of the ACM*, 40(1), pages 39 – 45, 1997.

[HJS00] W. Hoschek, J. Jaen-Martinez, A. Samar, H. Stockinger, K. Stockinger. Data Management in an International Data Grid Project. *IEEE/ACM International Workshop on Grid Computing 2000*, pages 77 – 90, 2000.

[HOTPAGE] The Hotpage project, https://hotpage.npaci.edu.

[HPF99] R. Housley, W. Polk, W. Ford, D. Solo. *Internet X.509 Public Key Infrastructure Certificate and CRL Profile,* 1999.

[ITSEC] *Information Technology Security Evaluation Criteria (ITSEC), provisional harmonized criteria version 1.2.* European Union.

[ITSEC-CC-1] *UKSP 14 Addendum: EAL1 Delta-evaluation.* EAL1 Delta Working Programme, Issue 2.B.

[ITSEC-CC-2] *UKSP 14 Addendum: EAL2 Delta-evaluation.* EAL2 Delta Working Programme, Issue 2.B.

[ITSEC-CC-3] *UKSP 14 Addendum: EAL3 Delta-evaluation.* EAL3 Delta Working Programme, Issue 2.B.

[ITSEC-CC-4] *UKSP 14 Addendum: EAL4 Delta-evaluation.* EAL4 Delta Working Programme, Issue 2.C.

[ITSEM] *Information Technology Security Evaluation Manual (ITSEM), version 1.0.* European Union.

[JAVA] Sun Java Technology, http://java.sun.com/.

[JMT98] W. Johnston, S. Mudumbai, M. Thompson. Authorization and Attribute Certificates for Widely Distributed Access Control. *7th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*, pages 340 – 345, 1998.

[JÜR02] J. Jürjens. Using UMLSec and Goal Trees for Secure Systems Development. *ACM Symposium on Applied Computing 2002*, pages 1026 – 1030, 2002.

[JÜR04] J. Jürjens. *Secure Systems Development with UML.* Springer, 2004.

[KHK04] B. Kim, S. Hong, J. Kim. Ticket-based Fine-grained Authorization Service in the Dynamic VO Environment. *ACM Workshop on Secure Web Services 2004*, pages 29 – 36, 2004.

[KKH03] S. Kim, J. Kim, S. Hong, S. Kim. Workflow-based Authorization Service in Grid. *IEEE/ACM International Workshop on Grid Computing 2003,* pages 94 – 100, 2003.

[KTB05] N. Kanaskar, U. Topaloglu, C. Bayrak. Globus Security Model for Grid Environment. *ACM SIGSOFT Software Engineering Notes*, 30(6), pages 1 – 9, 2005.

[LDAP] T. Howes. The Lightweight Directory Access Protocol: X.500 Lite. CITI Technical Report 95-8, University of Michigan.

[LK02] M. Lorch, D. Kafura. Supporting Secure Ad-hoc User Collaboration in Grid Environments. *IEEE/ACM International Workshop on Grid Computing 2002*, pages 181 – 193, 2002.

[MRA] *Mutual Recognition Agreement of Information Technology Security Evaluation Certificates - Version 2.0.* Management Committee of Agreement Group, http://www.cesg.gov.uk/site/iacs/itsec/media/formal-docs/MRA99.pdf.

[MYGRID] The myGrid project, www.mygrid.org.uk.

[NJD02] N. Nagaratnam, P. Janson, J. Dayka, A. Nadalin, F. Siebenlist, V. Welch, I. Foster, S. Tuecke. The Security Architecture for Open Grid Services. GGF Specification, 2002.

[NR05] S. Naqvi, M. Riguidel. Grid Security Services Simulator (G3S) – A Simulation Tool for the Design and Analysis of Grid Security Solutions. *1st International Conference on e-Science and Grid Computing*, pages 421 – 428, 2005.

[NWC04] M. Niinimaki, J. White, W. Cerff, J. Hahkala, T. Niemi, M. Pitkanen. Using Virtual Organizations Membership System with EDG's Grid Security and Database Access. *15th International Workshop on Database and Expert Systems Applications*, pages 517 – 522, 2004.

[OGSI] Open Grid Services Infrastructure (OGSI) - Version 1.0.

[OPENSSL] Open SSL project, http://www.openssl.org/.

[PERMIS] The PERMIS project, www.permis.org.

[PWF02] L. Pearlman, V. Welch, I. Foster, C. Kesselman, S. Tuecke. A Community Authorization Service for Group Collaboration. 3rd *IEEE International Workshop on Policies for Distributed Systems and Networks*, pages 50 – 59, 2002.

[RES04] C. Resch. Designing an Information Security System. *IEEE Workshop on Information Assurance 2004*, pages 449 – 450, 2004.

[RFC3281] RFC3281: *An Internet Attribute Certificate Profile for Authorization.* http://www.ietf.org/rfc/rfc3281.txt.

[RJS01] D. Roure, N. Jennings, N. Shadbolt. Research Agenda for the Semantic Grid: A Future e-Science Infrastructure. Report commissioned for EPSRC/DTI core e-Science Programme, 2001.

[RJS02] D. Roure, N. Jennings, N. Shabolt. The Semantic Grid: A Future e-Science Infrastructure. 2002.

[RM01] A. Rajasekar, R. Moore. Data and Metadata Collections for Scientific Applications. *European High Performance Computing Conference 2001*, pages 72 – 80, 2001.

[SAML] OASIA Security Assertion Markup Language (SAML), http://www.oasis-open.org/committees/security/#documents.

[SDSC] SDSC GridPort Toolkit, http://gridport.npaci.edu.

[SEMAN] W3C Semantic Web Activity Statement, http://www.w3c.org/2001/sw/Activity.

[SG03] T. Sandholm, J. Gawor. Globus Toolkit 3 Core – A Grid Service Container Framework. Beta draft, June 2003.

[SM1] *Information Technology – Code of Practice for Information Security Management.* ISO/IEC 17799.

[SM2] *Information Security Management Systems – Specification with Guidance for Use.* BS 7799.2.

[SMI01] R. Smith. Cost Profile of a Highly Assured, Secure Operating System. *ACM Transactions on Information and Systems Security*, 4(1), pages 72 – 101, 2001.

[SOAP] SOAP, http://www.w3.org/TR/SOAP/.

[SRE02] S. Shum, D. Roure, M. Eisenstadt, N. Shadbolt, A. Tate. CoAKTinG: Collaborative Advanced Knowledge Technologies in the Grid. *2nd Workshop on Advanced Collaborative Environment*, pages 21 – 28, 2002.

[SS75] J. Salzer, M. Schroeder. The Protection of Information in Computer Systems. *Proceedings of the IEEE*, 63(9), pages 1278 – 1308, 1975.

[SSE-CMM] *Information Technology - Systems Security Engineering - Capability Maturity Model (SSE-CMM®).* ISO/IEC 21827.

[SSL] Introduction to SSL, http://developer.netscape.com/docs/manuals/security/sslin/.

[SSLEAY] SSLeay project, http://www.columbia.edu/~ariel/ssleay/.

[SSW05] A. Stell, R. Sinnott, J. Watt. Comparison of Advanced Authorization Infrastructures for Grid Computing. *19th International Symposiums on High Performance Computing Systems and Applications*, pages 195 – 201, 2005.

[SWT02] F. Siebenlist, V. Welch, S. Tuecke, I. Foster, N. Nagaratnam, P. Janson, J. Dayka, A. Nadalin. OGSA Security Roadmap. GGF Specification, 2002.

[SXACML] The Sun's XACML Implementation project, http://sunxacml.sourceforge.net/.

[TCSEC] *Trusted Computer System Evaluation Criteria.* NCSC (5200.28-STD).

[TEM03] M. Thompson, A. Essiari, S. Mudumbai. Certificate-based Authorization Policy in a PKI Environment. *ACM Transactions on Information and System Security*, 6(4), pages 566 – 588, 2003.

[UDDI] UDDI, http://www.uddi.org.

[UML] OMG Unified Modeling Language specification, version 1.5, March 2003.

[UMLSEC] J. Jürjens. UMLSec: Extending UML for Secure Systems Development. *UML2002 – The Unified Modeling Language*, pages 412 – 425, 2002.

[UNICORE] The UNICORE project, http://www.unicore.org.

[VIE05] J. Viega. Building Security Requirements with CLASP. *ACM SIGSOFT Software Engineering Notes*, 30(4), pages 1 – 7, 2005.

[WS] Web Services, http://www.w3c.org/2002/ws.

[WSDL] WSDL, http://www.w3.org/TR/wsdl.

[WSRF] From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring & Evolution – Version 1.1.

[WSSEC] Security in a Web Services World: A Proposed Architecture and Roadmap, http://www-106.ibm.com/developerworks/library/ws-secmap/.

[WS03] V. Welch, F. Siebenlist. GT3 Grid Security Infrastructure Overview, 2003.

[XACML] OASIS eXtensible Access Control Markup language (XACML), http://www.oasis-open.org/committees/xacml.

[XMLDSIG] XML Digital Signature Specification, http://www.w3.org/TR/xmldsig-core/.

[X.509] ITU-T X.509 (2000): *The Directory: Authentication Framework.*

[ZBP05] X. Zhao, K. Borders, A. Prakash. SVGrid: A Secure Virtual Environment for Untrusted Grid Applications. *3rd Workshop on Middleware for Grid Computing*, pages 85 – 90, 2005.