PRIVACY-PRESERVING ATTRIBUTE-BASED ACCESS CONTROL IN A GRID

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

By

SANG MORK PARK
M.E., Air Force Institute of Technology, 2002
B. E., Air Force Academy, Republic of Korea, 1990

2010
Wright State University

Soon M. Chung, Ph.D.
Dissertation Director

Arthur Ardeshir Goshtasby, Ph.D.
Director, Computer Science and Engineering Ph.D. Program

Andrew T. Hsu, Ph.D.
Dean, School of Graduate Studies

Committee on Final Examination

Soon M. Chung, Ph.D.

Nikolaos Bourbakis, Ph.D.

Yong Pei, Ph.D.

Xinhui Zhang, Ph.D.

Chansu Yu, Ph.D.
Professor, Cleveland State University
ABSTRACT

Park, Sang Mork. Ph.D., Department of Computer Science and Engineering, Wright State University, 2010, Privacy-Preserving Attribute-Based Access Control in a Grid.

A Grid community is composed of diverse stake holders, such as data resource providers, computing resource providers, service providers, and the users of the resources and services. In traditional security systems for Grids, most of the authentication and authorization mechanisms are based on the user’s identity or the user’s classification information.

If the authorization mechanism is based on the user’s identity, fine-grained access control policies can be implemented but the scalability of the security system would be limited. If the authorization mechanism is based on the user’s classification, the scalability can be improved but the fine-grained access control policies may not be supported.

We developed an enhanced version of the Community Authorization Service (CAS) which supports centralized, fine-grained access control by managing the memberships, service types, resource objects and security policies of a Virtual Organization (VO) [13, 28]. The current CAS provides fundamental solutions regarding user privacy, authentication and authorization, but it has some limitations due to its centralized management of the security policies of a VO [8], in terms of scalability, flexibility and interoperability. We enhanced the CAS to support diverse security requirements within a dynamic Grid environment by enabling the CAS server to publish a proxy certificate embedding additional attributes of users. It allows the service
providers to support customized services by analyzing the attributes of users and security policies.

Previous researches on privacy-preserving in a Grid have focused on protecting the data stored in a data server and on securing the communication to protect exchanged data. The issue of preserving the privacy of users has not been a major issue in the security domain. However, as on-line transactions prevail and diverse user attributes are required for authorization decision, the privacy-preserving becomes an important issue.

Attribute-Based Access Control (ABAC) employs multiple attributes for authorization decision, which enables the security system to be flexible, interoperable, and multifunctional. However, ABAC has disadvantages with regard to privacy-preserving because it requires the circulation of the user attributes which can increase the risk of privacy violation.

To enhance the privacy-preserving capability of ABAC in a Grid, we developed an attribute release control mechanism to publish an optimal set of attributes that are essential to access a desired resource (or service), while exposing least amount of sensitive user information. To facilitate the selection of an optimal set of attributes, we also developed Security Policy Publication Service (SPPS) which retrieves the access condition from the access control policies in eXtensible Access Control Markup Language (XACML) [26] and converts it into a Disjunctive Normal Form (DNF) of attributes. We modified the Shibboleth Identity Provider [19] and GridShib [23, 42] for the implementation of our privacy-preserving ABAC, and the performance analysis shows that the overhead of the proposed system is very small.
TABLE OF CONTENTS

ABSTRACT .......................................................................................................................................................... iii

TABLE OF CONTENTS ........................................................................................................................................ v

LIST OF FIGURES ............................................................................................................................................... vii

ACKNOWLEDGEMENT ...................................................................................................................................... viii

1. INTRODUCTION ........................................................................................................................................... 1

2. BACKGROUND .................................................................................................................................................. 5
   2.1. Identity Federation Framework ............................................................................................................. 5
   2.2. Shibboleth Identity Provider .................................................................................................................. 7
   2.3. GridShib Interface for Shibboleth .......................................................................................................... 8
   2.4. Globus Toolkit ....................................................................................................................................... 8
   2.5. Security Assertion Markup Language (SAML) .................................................................................... 10
   2.6. Attribute-Based Access Control ........................................................................................................ 11
   2.7. eXtensible Access Control Markup Language (XACML) ................................................................ 12
   2.8. Attribute Release Policy ..................................................................................................................... 13

3. ENHANCED COMMUNITY AUTHORIZATION SERVICE (CAS) CERTIFICATE FOR METADATA-BASED ACCESS CONTROL (MBAC) 15
   3.1. CAS Overview ....................................................................................................................................... 16
       3.1.1. Components of the CAS ................................................................................................................ 16
       3.1.2. Operational Steps of the CAS ....................................................................................................... 17
       3.1.3. Limitation of the Current CAS .................................................................................................... 19
       3.1.4. Security Assertion Markup Language (SAML) and X.509 Public Key Infrastructure (PKI) ......... 20
   3.2. Motivation for Enhancing the CAS ....................................................................................................... 22
       3.2.1. Metadata-Based Access Control (MBAC) .................................................................................... 23
       3.2.2. Advantages of the Enhanced CAS ............................................................................................... 24
   3.3. Embedding Metadata in the CAS Proxy Certificate ............................................................................. 25
       3.3.1. Modification of the CAS Policy Database in the Globus Toolkit .............................................. 26
       3.3.2. Modification of the CAS Server .................................................................................................. 26
   3.4. Performance Analysis ........................................................................................................................... 28
   3.5. Summary of Enhanced CAS ................................................................................................................ 29
4. PRIVACY-PRESERVING ATTRIBUTE-BASED ACCESS CONTROL (ABAC)...........................................................................................................31
   4.1. Introduction...............................................................................................31
   4.2. Proposed System Architecture ..............................................................33
   4.3. Security Policy Publication Service (SPPS)............................................34
       4.3.1. Definition of the SPPS Interface......................................................37
       4.3.2. Implementation of the SPPS ...............................................................39
       4.3.3. Deployment Configuration of the SPPS..........................................42
       4.3.4. Compile Java Programs and Organize the Files ..............................43
       4.3.5. Deploy the SPPS in Web Service Container ....................................43
       4.3.6. Conversion of an Access Condition to a Disjunctive Normal Form
               (DNF) ...............................................................................................44
   4.4. Privacy-Preserving Shibboleth Identity Provider ....................................45
       4.4.1. Selection of an Optimal Set of Attributes .......................................46
   4.5. Performance Analysis...............................................................................47
       4.5.1. Overhead of SPPS ............................................................................48
       4.5.2. Overhead of Privacy-Preserving Shibboleth Identity Provider .........49
   4.6. Summary of Privacy-Preserving ABAC System .....................................50

5. CONVERSION OF PROPOSITIONAL LOGIC FORMULA INTO DNF BY
   USING A GRAPH .................................................................................52
   5.1. Introduction...............................................................................................52
   5.2. Motivation of DNF Research ................................................................54
   5.3. Background .............................................................................................55
       5.3.1. Conjunctive and Disjunctive Normal Forms .....................................55
       5.3.2. Tree vs. Graph for Representing Logic Formulas Forms ..................56
   5.4. DNF Conversion Algorithm ...................................................................57
       5.4.1. Represent a Propositional Logic Formula with a Graph ....................57
       5.4.2. Graph Traversal to Obtain the Conjunctive Clauses .......................59
       5.4.3. Prune Redundancy and Remove Contradiction .................................59
       5.4.4. Selection of the Optimal Combination of Attributes ..........................60
   5.5. Performance Issue ..................................................................................61
   5.6. Summary of DNF Conversion ...............................................................61

6. CONCLUSION ...........................................................................................63

BIBLIOGRAPHY .........................................................................................65
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Identity Federation Framework (Push Model)</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>Shibboleth Identity Provider</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>Interactions between a Client and the Shibboleth Identity Provider</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>Globus Toolkit 4 Architecture</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>Operational Steps of the CAS</td>
<td>18</td>
</tr>
<tr>
<td>6</td>
<td>Content of the Current CAS Certificate</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>Contents of the Modified CAS Certificate</td>
<td>27</td>
</tr>
<tr>
<td>8</td>
<td>Time for Creating <code>&lt;AuthorizationDecisionStatements&gt;</code> Element</td>
<td>29</td>
</tr>
<tr>
<td>9</td>
<td>Proposed System Architecture</td>
<td>34</td>
</tr>
<tr>
<td>10</td>
<td>SPPS Architecture</td>
<td>36</td>
</tr>
<tr>
<td>11</td>
<td>SPPS Interface Definition in WSDL</td>
<td>38</td>
</tr>
<tr>
<td>12</td>
<td>XACML Request Context</td>
<td>40</td>
</tr>
<tr>
<td>13</td>
<td>XACML Policy</td>
<td>41</td>
</tr>
<tr>
<td>14</td>
<td>WSDD File for the SPPS Deployment</td>
<td>42</td>
</tr>
<tr>
<td>15</td>
<td>JNDI Configuration for the SPPS Deployment</td>
<td>43</td>
</tr>
<tr>
<td>16</td>
<td>Globus Web Service Container Log</td>
<td>44</td>
</tr>
<tr>
<td>17</td>
<td>Overhead of SPPS</td>
<td>48</td>
</tr>
<tr>
<td>18</td>
<td>Overhead of Shibboleth Identity Provider (IdP)</td>
<td>50</td>
</tr>
<tr>
<td>19</td>
<td>Tree vs. Graph</td>
<td>57</td>
</tr>
<tr>
<td>20</td>
<td>Node Representation</td>
<td>58</td>
</tr>
<tr>
<td>21</td>
<td>Graph Representation of the Propositional Logic Formula (2)</td>
<td>58</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENT

There are a number of people without whom this research could not have been completed. First of all, I would like to express my gratitude and respect to my advisor, Dr. Soon Chung, for his guidance and dedication to improve my research. Dr. Chung has provided me not only academic expertise but also practical guidance for my career improvement. Thank you very much Dr. Chung for everything you have done for me.

I also would like to thank all my committee members, Dr. Nikolaos Bourbakis, Dr. Yong Pei, Dr. Xinhui Zhang and Dr. Chansu Yu, for their helpful comments and feedbacks. I would also like to thank the Korea Air Force, the Dayton Area Graduate Studies Institute (DAGSI), and the Department of Computer Science and Engineering at Wright State University for supporting my study.

I also thank Dr. Vineela Muppavarapu who had been a senior at the Database Research Laboratory. Her technical suggestions contributed a lot to this research.

I would like to thank my parents, Mr. Seung T. Park, Mrs. Bun Y. Jeong, a brother in heaven, and four sisters. Finally, I would like to thank my loving wife Seung-Yeon and our son Jae-Min for their support and endurance during my Ph.D. study.
1. INTRODUCTION

Grid computing is currently being integrated into the Internet supported Web service domain. A Virtual Organization (VO) is a basic entity of Grid computing. It is composed of the resources, data, and devices presented by different physical organizations. A VO dynamically collects and integrates computer resources from logically and physically distributed organizations such as schools, companies, and laboratories, and uses the resources in order to overcome the limitation of individual computing systems.

Since the resources and members are various and heterogeneous in a VO, it is a very important but difficult task to build a reliable, effective and efficient security system. One of the most important requirements for building the security system is providing an access control mechanism that can comply with the diverse requirements of different participants and can deal with the heterogeneous platforms and operating systems in the VO.

When an access control system makes an authorization decision for a user to access a resource or a service, it checks either the identity or the classification of the user. The access control system based on user identities can support fine-grained access control policy implementation, but the system’s scalability degrades significantly as the number of users, resources, and allowable actions increase, because of the mapping scheme for authorization decision. The number of policy statements will increase exponentially as the number of users, resources and actions grows.
In order to address the scalability problem, user classification is used for access control. This method classifies the users into different groups, and the groups are mapped to the resources and actions. Some access control schemes such as Mandatory Access Control (MAC) and Role-Based Access Control (RBAC) [7] are based on the user classification. However, user classification information can not represent individual users clearly. If the number of groups is increased to address this granularity issue, it will cause the scalability problem.

Attribute-Based Access Control (ABAC) [2, 21, 41] scheme can address the issues of both granularity and scalability. Since ABAC employs multiple user attributes, it can support fine-grained access control policy implementation, while providing the scalability in performance and administration.

There is another approach to deal with the scalability problem. In a VO, there are numerous participants such as service providers and users. If a user wants to access multiple resources presented by different resources providers, the user has to possess multiple accounts on the resource provider. This will require the resource and service providers to manage the information of the users, which will cause significant security administration overhead for the local service providers. In order to avoid the security overhead due to redundant membership management in local service providers, VO-level centralized access control mechanisms are developed. The centralized access control mechanisms, such as Community Authorization Service (CAS) [27], Virtual Organization Membership Service (VOMS), and Enterprise Authorization and Licensing Service (EALS) [6] are widely deployed in different Grids, and provide scalable VO-level membership management.
However, the scalability of the VO-level security system can also degrade if it uses individual user identity. Anil et al. [30] addressed the scalability problem of VO-level security system by adopting the RBAC in CAS. But the RBAC may not be able to provide sufficient scalability for federated, large-scale VOs and Web service communities which provide services to other organizations and communities.

To address this scalability issue of a large-scale VO, the Identity Federation Framework is introduced. The framework depends on the trusted relationship between the members of VO. If a service provider trusts another service provider which stores a user’s profile and can assure the membership of the user, the user profile can be shared between them, which enables the user to access any resources and services in the VO without redundant registration or a centralized VO-level security system.

ABAC scheme with an identity federation framework can solve the issues of scalability and the granularity of policy information, because the local service providers can provide user identity and attributes to other members in the VO who are within the same trust domain with the service provider. The user information can be used for the authorization decisions by different service providers.

Even though the ABAC scheme within the identity federation framework has advantages regarding scalability and granularity, it has disadvantages in privacy-preserving. Since it requires the circulation of user identity information and other user attributes, it can violate the privacy of the users. The privacy-preserving of the users becomes more significant as the scope of VO increases, because more sensitive user attributes can be circulated and the possibility of malicious participant’s intervention increases.
In this research, we developed two enhanced mechanisms for the security systems in a Grid, in terms of scalability, flexibility, interoperability and privacy-preserving.

- The current Community Authorization Service (CAS) has some limitations due to its centralized management of the security policies of a virtual organization. We developed an enhanced CAS to improve its flexibility and interoperability. The enhanced CAS can issue a proxy certificate which can deliver any kind of semantic and contextual information to support diverse access control mechanisms, such as Role-Based Access Control (RBAC), Metadata-Based Access Control (MBAC), Semantic Access Control (SAC) and Attribute-Based Access Control (ABAC).

- The ABAC is a flexible, interoperable and scalable access control scheme. However, it has disadvantage in terms of privacy protection due to the distribution of user attributes. We developed an attribute release control mechanism to publish an optimal set of attributes that are essential to access a desired resource (or service), while exposing least amount of sensitive user information. We also developed Security Policy Publication Service (SPPS) which retrieves the access condition from the access control policies in eXtensible Access Control Markup Language (XACML) [26] and converts it into a Disjunctive Normal Form (DNF) of attributes. We modified the Shibboleth Identity Provider [19] and GridShib [23, 42] for the implementation of our privacy-preserving ABAC, and the performance analysis shows that the overhead of the proposed system is very small.
2. BACKGROUND

This chapter introduces all the major concepts, standards, protocols, and interfaces used in this research. Our proposed security system is implemented using the Shibboleth Identity Provider, GridShib, and Globus Toolkit 4. These open source software systems are built on numerous Web services and Web communication standards, protocols, and interfaces.

2.1. Identity Federation Framework

The identity federation framework refers to a security system infrastructure that shares user identities between different security domains. Most valuable resources in Grid environment are protected by authentication and authorization systems, and their processes depend on user identities that are shared by the participants of the VO within an identity federation framework. In a security system without an identity federation infrastructure, every resource provider has to manage the user accounts for its authorization decisions, or there should be a centralized authentication and authorization system. Both the local membership management system and the centralized authentication and authorization system are not scalable as the VO expands.

The identity federation framework can solve the scalability problem because each Identity Provider manages the membership of its local users, and the local identity information repositories can be shared by the participants of the VO. Thus, the VO does not have to consider the administration of the mapping between users and resources for authorization decision.
An advantage of the identity federation and trust federation among different participants in a VO is that we can support the single sign-on [3].

Figure 1. Identity Federation Framework (Push Model)

Figure 1 illustrates a high level view of the identity federation framework in a push model, composed of Shibboleth Identity Provider, GridShib interface for Shibboleth, and Globus Toolkit [15]. When a client requests an attribute assertion to the Shibboleth Identity Provider, it authenticates the client and retrieves the attributes from the identity information repository according to the attribute release policies, and then returns the attribute assertion to the client in the form of Security Assertion Markup Language (SAML) [24, 25]. The client submits the assertion to the resource provider in order to access its resource.
2.2. Shibboleth Identity Provider

The Shibboleth Identity Provider is a component of Shibboleth and performs the identity brokerage by publishing attribute assertions for the distribution of identity information. It contains four functional components: authentication authority, attribute authority, single sign-on service, and inter-site transfer service [4], as shown in Figure 2.

![Figure 2. Shibboleth Identity Provider](image)

Authentication authority deals with the authentication processes of the users who registered to a local Identity Provider and issues an authentication assertion. The Shibboleth Identity Provider supports diverse authentication mechanisms to cope with different security requirements.

Attribute authority handles the SAML protocol binding and issues attribute assertions to the assertion consumers. Only the attributes allowed by the Attribute Release Policy (ARP) can be released.
Single sign-on service is a HTTP resource that receives and processes authentication requests presented by the assertion consumers. It can return an error message or redirect the assertion request to the inter-site transfer service.

Inter-site transfer service issues HTTP responses with the collaboration of authentication authority.

2.3. GridShib Interface for Shibboleth

GridShib interface for Shibboleth enables the user’s client program to access the Shibboleth Identity Provider. Figure 3 shows the interactions between a client and the Shibboleth Identity Provider. The client presents the user’s distinguished name for authentication, then the attribute authority maps the user’s distinguished name into a local user name which is used as the key for the retrieval of user attributes.

![Figure 3. Interactions between a Client and the Shibboleth Identity Provider](image)

2.4. Globus Toolkit

The Globus Toolkit [15] is an open source software package supporting the construction of a Grid and the implementation of diverse Web services in a Grid. It is
composed of predefined Web service components, non-Web service components, libraries and APIs as shown in Figure 4, and they provide security, data management, and service management for a Grid.

Grid Security Infrastructure (GSI) [35], employed by the Globus Toolkit, supports secure communication between the participants in a Grid, single sign-on for users in different organizations, and cross-organizational security system implementation. GSI uses a certificate for mutual authentication and secure communication, and the certificate
is issued by a Certificate Authority (CA) trusted by the participants of the Grid community.

2.5. Security Assertion Markup Language (SAML)

SAML [24, 25], developed and managed by the Security Services Technical Committee (SSTC) of Organization for the Advancement of Structured Information Standards (OASIS), is an XML based standard that supports secure exchange of security information between the participants of a VO.

SAML is composed of four major components: assertions, protocols, bindings and profiles [25]. SAML can publish and exchange three types of assertions: authentication assertion, attribute assertion, and authorization assertion. Authentication assertion describes which Identity Provider has authenticated the specified subject and informs how and when the authentication is performed. It also contains constraints of the assertion, such as the effective period of the assertion. An attribute assertion accommodates the properties and values of specific subject’s attributes. A typical attribute assertion contains the attributes associated with the identity of the subject. However, it can also contain diverse information to satisfy different security policy requirements. Authorization assertion informs the authorized actions on different resources to a specific subject. Protocols define the way how communications between Identity Providers and assertion consumers are performed. Bindings denote how the SAML request and response messages are enveloped.

The SAML messages are exchanged by the Simple Object Access Protocol (SOAP) [39] and HTTP support. SAML Profile describes how the other components are
combined together by defining constraints and extensions in order for a particular application to use SAML.

SAML has advantages in the flexibility of application design, cost reduction for security system administration, and protecting user privacy. Security system designers can manipulate the current SAML protocols and specifications in order to build customized security systems.

SAML architecture assumes distributed management of attributes by local Identity Providers, and a federation of authentication and authorization systems of local resource providers, which can reduce the cost for security systems.

2.6. Attribute-Based Access Control

The Attribute-Based Access Control (ABAC) has advantages in terms of flexibility, interoperability, scalability, and fine-grained access control capability, compared with the traditional access control schemes, such as mandatory access control (MAC), discretionary access control (DAC), and role-based access control (RBAC) [7].

The authorization decisions in traditional access control schemes depend only on the classification of users based on their identities. However, the authorization decision in ABAC depends on the attributes of the users which can provide diverse information of users. Since ABAC presumes the availability of multiple attributes for security operations, it is capable of implementing any type of access control schemes. The attributes retrieved for access control can accommodate the user’s properties required for MAC, DAC and RBAC, so it makes ABAC flexible and interoperable with different access control schemes. In addition to the user attributes, SAML assertions can carry
additional information for access control, such as conditions and constraints. This allows ABAC to implement flexible and fine-grained access control policies. The distributed management of attribute repository and security policy in ABAC provides scalability.

2.7. eXtensible Access Control Markup Language (XACML)

XACML is an OASIS standard language describing access control policies uniformly across different security domains [26].

XACML defines the following main components to represent the policies:

(1) A <PolicySet> contains a set of access control policies or other policy sets.

(2) A <Policy> represents an access control policy described through a set of rules.

(3) A <Rule> represents an access rule or permission.

An XACML <PolicySet>, <Policy> or <Rule> may contain a <Target> element. A <Target> element specifies a set of subjects, resources, actions and environments to which the <PolicySet>, <Policy> or <Rule> is applied [26].

In XACML authorization model, a Policy Information Point (PIP) releases attribute values related to the subject (such as a user, an application or a Grid service), the resource and the environment. a Policy Administration Point (PAP) manages the policies and policy sets, and makes them available to a Policy Decision Point (PDP). The PDP returns the authorization decision, such as “permit” or “deny”, and a Policy Enforcement Point (PEP) executes the decision of the PDP by either performing or denying the client’s request.
2.8. Attribute Release Policy

Attribute Release Policy (ARP) [18] scheme is a filtering mechanism designed to prevent the circulation of sensitive information by prohibiting Identity Providers to publish the attributes of sensitive information. The ARP is a set of policies describing the permitted or prohibited attributes with regard to the users and member sites of a VO. The user ARP is applied to the individual users in the VO, and the site ARP is applied to the member sites in the VO.

The ARP can be used as a tool for privacy preservation within an identity federation framework. However, there are two problems in applying the current ARP scheme to a complex security system because it employs a simple filtering mechanism.

The first problem of the current ARP scheme is that it cannot assign different levels of sensitivity for different attributes. Different attributes may have different levels of sensitivity. For example, sensitivity of a student’s grade is greater than that of a course taken by the student. The current ARP scheme can make only true or false decision for attribute release.

The second problem of the current ARP scheme is that it cannot balance the privacy protection and the secure access to resources. Shibboleth Identity Providers will present the attributes which are not prohibited by an ARP. Some of the attributes included in assertions may be unnecessary for the client to access certain resources. Then, those unnecessary attributes would increase the risk of privacy violation as well as the communication load without any benefit. If an attribute prohibited to be released by an ARP is required to access to a resource, the client will never be able to use it. There may
be cases that a client is willing to sacrifice certain portion of his/her privacy in order to use certain resources.

In order to solve the problems of current ARP scheme, an improved attribute release control scheme is designed. Our new scheme enables the Identity Providers to consider the sensitivity level of each attribute, to prevent unnecessary attributes from being published, and to optimize the selection of attributes for privacy preservation.
3. ENHANCED COMMUNITY AUTHORIZATION SERVICE (CAS) CERTIFICATE FOR METADATA-BASED ACCESS CONTROL (MBAC)

Implementing scalable and interoperable authentication and authorization schemes is an important issue in Grid computing, especially when resource sharing is highly demanded within a VO [27, 28].

The Globus Toolkit [15] is a prominent toolkit enabling the formulation of a Grid computing environment, and it adopted the Grid Security Infrastructure (GSI) [35] as the security basis. The Globus Toolkit also supports the Community Authorization Service (CAS), a centralized administration system for the authentication and authorization policies of VOs. The CAS can reduce the burden of the local service and resource providers by managing the access control information, however it has some limitations in scalability and flexibility. As the number of resource providers and users increases in a VO, the complexity of the access control management may become very high. When a VO has $P$ users and $Q$ resources, and there are $R$ kinds of permissions for each resource, $P \times Q \times R$ entries of security policy are required to provide fine-grained access control with the CAS [27].

In this chapter, we propose an enhanced version of CAS to improve the scalability and flexibility, as well as the interoperability between VOs. When a user wants to access a resource within a VO, it requests a CAS proxy certificate. Then, the CAS server publishes an X.509 proxy certificate based on the identity of the user and the security policies stored in the policy database [28]. The current CAS embeds the user identity, resource objects and the actions permitted to the user within the
<AuthorizationDecisionStatement> element of the Security Assertion Markup Language (SAML) [24] assertion. We modified CAS to publish a proxy certificate embedding additional metadata of the user. To comply with the SAML specifications, <Attribute> element is created and the user’s metadata is embedded in the <AuthorizationDecisionStatement> element.

As the modified CAS certificate carries additional attributes of users, it can support various access control schemes, such as the Metadata-Based Access Control (MBAC) [43] and the Semantic Access Control (SAC) [44], which are emerging schemes in the Web Service security domain. The enhanced CAS involves some overhead to embed and lookup additional metadata in its certificates, but it is quite negligible.

3.1. CAS Overview

The CAS was developed to provide tools for centralized security administration in a VO. It is a collection of software packages which enables a VO to design and manage its own set of security policies within a static trust domain. It adopted the Public Key Infrastructure (PKI) for secure communication, and embodied modern security schemes such as secure authentication and authorization, single-sign-on, and delegation [8, 13, 28].

3.1.1. Components of the CAS

- CAS client: A CAS client is a service consumer (i.e., user) who requests the CAS certificate, stores the published certificate in local storage space, and utilizes the certificate by wrapping it with the service request arguments. A user’s private proxy
certificate, published by the Globus Toolkit, is required to request a CAS service. It provides the identity information of the user. [14]

- **CAS server**: A CAS server is a service provider who furnishes a VO level access control administration service. It reads the user's service request, authenticates the user, and retrieves the authorization policy data from the CAS policy database. Based on the retrieved security policy data, the CAS server publishes an X.509 formatted CAS proxy certificate, and returns the certificate to the user [8].

- **CAS policy database**: A CAS policy database stores the subjects, resource objects and access control policies of the VO. It can support diverse access control granularities such as individual identity-based access control and group-based access control. There are four types of data entries in the database. The *users* entry contains the identity information of the members of the VO, and the *action* entry stores the types of actions and services permitted to the members. The *resource* entry contains the identity information of the resources which are available in the VO, and the *policy* entry stores the permitted actions and resources [8, 13].

### 3.1.2. Operational Steps of the CAS

The CAS follows GSI specifications which employ the proxy certificate-based access control mechanism. Figure 5 briefly illustrates the operational flow of the CAS and how a proxy certificate is used for a service within a Grid environment.
A user who wants a Grid service submits a CAS certificate publication request to the CAS server with its private certificate and option parameters. In order to issue the request, the user is required to possess a private proxy certificate which can be trusted within the VO.

When a certificate request is presented, the CAS server authenticates the user, queries the CAS policy database to retrieves the services, objects and actions authorized to the user [8]. The retrieved information is assembled into a SAML assertion, and an X509 proxy certificate is issued with the assertion and then returned to the user.

The user uses the certificate by wrapping it with the service request arguments.

When a service provider receives a service request, it authenticates the user, identifies the authorized resources and actions, and then provides the requested service based on the local security policies.
3.1.3. Limitation of the Current CAS

The current CAS server publishes a proxy certificate containing the identity of registered users, permitted resources and actions based on the content of the CAS policy database. By utilizing the proxy certificate, diverse security policies can be applied within a VO. However, only static security schemes can be supported because the CAS proxy certificate can carry predetermined security policy properties.

Figure 6. Content of the Current CAS Certificate

Figure 6 illustrates the contents of the current CAS proxy certificate. It contains a set of authentication and authorization policy entries stored in the CAS policy database. The Decision and Resource attributes describe the accessibility of the <Subject> for a specific resource. The <Subject> element identifies who the permitted user is, and the
<Action> element shows the permitted resource and operations for the user. The certificate given in Figure 6 explains that the designated subject is permitted to access the files within the directory of FTPDirectoryTree|ftp://cspc/. With this example, we can see that the current CAS has a limitation in scalability because the security policies regarding each of the <Subject>, Resource and <Action> have to be established and managed individually.

The current CAS has a limitation in flexibility. A VO can be organized and disorganized dynamically based on its objective. The memberships and security policies of a VO also can be changed dynamically. As the current CAS assumes individual adoption of security policies on its memberships, changing the memberships and local security policies require the modification of relevant CAS policy database entries, and human intervention is required in its administration loop.

The current CAS is supposed to support only one VO-level security mechanism. Thus, it is not suitable for the security management of federated VO's even though they are within the same trust domain.

3.1.4. Security Assertion Markup Language (SAML) and X.509 Public Key Infrastructure (PKI)

The CAS follows the Security Assertion Markup Language (SAML) specifications for organizing its proxy certificate assertion. SAML is a standard for exchanging authentication and authorization decisions across different security policy enforcement points [10]. The Security Services Technical Committee of the Organization for the Advancement of Structured Information Standards (OASIS)
published the SAML specifications to provide a guidance for implementing single-sign-on and delegation schemes in the Grid security domain [24]. The core module of SAML is the assertion which supports authentication, authorization and attribute arrangement.

The SAML specifications support three defining types of assertions, which are authentication assertion, attribute assertion and authorization assertion, for access control decision [10]. Authentication assertion contains the user identity, the confirmation status of the user identity, the confirmation method, and the constraints of the proxy certificate containing the assertion. It informs us that a user was authenticated by an Identity Provider at a particular time with a particular authentication method, and the proxy certificate is valid only when the specified constraints are satisfied [10]. Attribute assertion holds the user attributes for further identity information provision. Authorization assertion contains the authorized user, the resource and the actions permitted to the user. It describes that a user <Subject> is authorized to perform an <Action> to given Resource [10]. The current CAS proxy certificate accommodates only Authentication and Authorization assertions.

X.509 PKI is a standard for exchanging and validating a public key, published by ITU-T [33, 37]. A root certificate, distributed from the organization's trusted certificate authority (CA), is the origin of trust endorsement between participants in the organization. The participants utilize the root certificate to assure the validity of their proxy certificate and to make the certificate delegate them. A certificate following X.509 standard contains the version, serial number, algorithm ID, issuer, validity, subject, subject’s public key information and some optional supplements [33]. A CAS proxy certificate contains the SAML assertion wrapped by X.509 standard.
3.2. Motivation for Enhancing the CAS

Let’s consider a case that a user tries to access multiple services within a VO by using a proxy certificate which contains only authentication and authorization information. If all the service providers adopt homogeneous security administration tools and they have access control policies regarding the user’s identity, the user may accomplish the goal without significant security disturbances. However, it may not be feasible to adopt homogeneous security mechanisms within a Grid environment.

The Globus Toolkit adopted the CAS to resolve the security issues with a condition that all the members of a VO agreed on the centralized management of security policies. However, as we discussed in the previous section, the current CAS has some limitations regarding scalability, flexibility and interoperability.

Pereira et al. [29] proposed to use Role-Based Access Control (RBAC) to reduce the administration overhead in the Data Grid using the CAS. They demonstrated that a RBAC scheme adopted to the CAS can improve the scalability with an example of Grid database services. Even though the RBAC mechanism can improve the scalability of the current CAS, we still have the flexibility and interoperability issues.

By enabling the CAS to embed the user’s metadata in its proxy certificate, we can support additional access control mechanisms, such as Metadata-Based Access Control (MBAC).
3.2.1. Metadata-Based Access Control (MBAC)

User identity is a key component for the access control in Grids. A combination of a user name and a password is the basis of the user identity for fine-grained access control; and the role of a user in Role Based Access Control (RBAC), the security label in Mandatory Access Control (MAC), and the delegated authority in Discretionary Access Control (DAC) are the bases of the user identity in coarse-grain access control schemes [8]. The identity-based access control mechanism has been widely used because of the simplicity in its implementation and application. However, it has limitations regarding scalability and interoperability. It may require a large amount of identity data to be mapped into access control policies, and it may be impractical to share the identities between different access control schemes.

In order to cope with the scalability and interoperability problems, Yague et al. [43] proposed Metadata-Based Access Control (MBAC). They demonstrated that MBAC can provide a great extent of flexibility and interoperability in dynamic and heterogeneous security domains. A certificate containing the user identity and supplemental attributes plays a key role in improving the flexibility and interoperability of the security system. The attributes to be included in the certificate can be selected based on the security requirements. They can be user identity, memberships, roles, security labels, delegation authorities, and so on.

Nowadays, many researchers try to utilize the semantic and contextual information carried by a secure certificate for access control, in order to improve the manageability over a large number of distributed resources and to enhance the interoperability between heterogeneous security domains [44]. Even though there is no
explicit authorization statement, we can validate the accessibility of a user by analyzing the semantic and contextual information of the user attributes.

Semantic Access Control (SAC) [44], Concept-level Access Control [31], Ontology-based Right Expression Language (OREL) [32], and eXtensible Access Control Markup Language (XACML) [1, 26] are proposed based on the idea of utilizing the metadata for the management of access control.

### 3.2.2. Advantages of the Enhanced CAS

In order to address the scalability, flexibility and interoperability problems of the current CAS, we modified the CAS server to publish a proxy certificate carrying user attributes. The attributes embedded in the certificate can reduce the security administration cost for the VO.

Normally, when a resource provider changes its local security policy, the CAS administrator should update the CAS policy database first to produce a proxy certificate reflecting the policy change. However, if a resource provider can parse the certificate and analyze the user attributes, local security policy change may not necessarily require the corresponding update on the CAS policy database.

The user attributes embedded in the CAS proxy certificate can make it interoperable in multiple VOs if they are within the same trust domain. The user attributes can also allow the CAS to support diverse access control mechanisms.

As the CAS is a built-in component of the Globus Toolkit for VO-level security administration, we can utilize the GSI specifications, employed by the toolkit, for implementing a secure metadata exchange mechanism. GSI specifications support secure
communication, X.509 certificate-based authentication, single-sign-on, delegation, and multi-layered security mechanisms [9, 16]. Shibboleth [19] is an attribute authorization service and is designed to provide user attributes to the service providers for access control. An Identity Provider (IdP) can publish a proxy certificate containing the user attributes to present it to a service provider. Service providers make a decision on the accessibility of a proxy certificate owner by analyzing the attributes contained in the certificate. Even though both Shibboleth and our enhanced CAS can produce the proxy certificates containing user attributes, the latter shows better performance because no additional network connections and software setups are required for the exchange of attributes. The assertion included in the CAS proxy certificate follows the SAML standard. This allows the certificate to comply with diverse SAML oriented security administration tools.

In the rest of this chapter, we focus on the modification of the CAS server to publish a certificate containing user attributes for the exchange of identity-related information. Development an access control scheme utilizing the semantic and contextual information of the user attributes is an on-going project.

3.3. Embedding Metadata in the CAS Proxy Certificate

In order to improve the usability of a CAS certificate, we modified the CAS server to publish a proxy certificate including supplemental attributes. The additional attributes in the certificate can be selected based on the purpose of the certificate. For example, let’s consider a case that a university student who is involved in a collaborative project between a university and a research organization. If the university and the
research organization have heterogeneous access control schemes, but they are within the same trust domain, the student may want to possess a certificate containing both academic attributes and research-related attributes. To implement this case, we modified the CAS policy database schema available in the Globus Toolkit, and also modified a part of the CAS server.

### 3.3.1. Modification of the CAS Policy Database in the Globus Toolkit

The CAS database system in the Globus Toolkit 4.0.4 has 13 tables, and they store information about users, resource objects, actions, and authorization policies. The `user_table`, having columns of `userNickname`, `subjectName` and `trustAnchorNickname`, provides the identities of users in the VO. Each user is uniquely identified by `subjectName` and `trustAnchorNickname`.

We modified the `user_table` by adding two additional columns: `academicAttributeName` and `researchAttributeName`, and they are linked to `academic_attribute_table` and `research_attribute_table`, respectively, as foreign keys. Both `academic_attribute_table` and `research_attribute_table` contain some relevant attributes.

### 3.3.2. Modification of the CAS Server

A CAS proxy certificate is published based on the parameters specified in the certificate request and the security policies stored in the CAS policy database. A user submits a certificate request with optional parameters and the user’s identity information,
which is included in the user’s private proxy certificate. The current CAS proxy certificate does not hold user attributes because they are not in the CAS policy database, and there is no section within the assertion that can accommodate those attributes. The CAS server software was modified to query the modified policy database, and to allow the user attributes to be inserted in the assertion of the proxy certificate when the user requested a certificate containing them.

```xml
<Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion" AssertionID="800a3c20-bd50-11dc-a21f-8f13cd01fccc" IssueInstant="2008-01-07T18:43:55Z" Issuer="...CN=Grid Simple CA" MajorVersion="1" MinorVersion="0"> ... <AuthorizationDecisionStatement Decision="Permit" Resource="FTPDirectoryTreeftp://cspc85.cs.wright.edu/home/spark/*"> <Subject> <NameIdentifier Format="#X509SubjectName" NameQualifier="O=Grid,OU=GlobusTest,OU=simpleCA-motive.cs.wright.edu,OU=cs.wright.edu,CN=Sang Park" O=Grid/OU=GlobusTest/OU=simpleCA-...cs.wright.edu/OU=cs.wright.edu/CN=... </NameIdentifier> <SubjectConfirmation> <ConfirmationMethod>... </ConfirmationMethod> </SubjectConfirmation> </Subject> <Action Namespace="cas">grantAll</Action> </AuthorizationDecisionStatement> <AssertionIDReference> <Attribute AttributeName="urn:mace:dir:attribute-def:current_program" AttributeNamespace="urn:mace:shibboleth:1.0:attributeNamespace:uri"> <AttributeValue>Ph.D.Student</AttributeValue> </Attribute> <Attribute> <Attribute AttributeName="..." AttributeNamespace="...">...</Attribute> </AssertionIDReference> </AuthorizationDecisionStatement> ...
```

**Figure 7. Contents of the Modified CAS Certificate**

There is an unused section within the `<AuthorizationDecisionStatement>` element. Based on the SAML specifications, the `<AssertionIDReference>` component is supposed to provide the confirmation information of the certificate’s trust domain. However, the
current CAS proxy certificate does not use it and leave it as a null value. We modified the CAS server to insert supplemental attributes into that section when a user requests it.

Figure 7 illustrates the modified CAS proxy certificate. We can see that a number of \(<\text{Attribute}>\) elements are inserted within the \(<\text{AssertionIDReference}>\) element. The values of the \(<\text{Attribute}>\) elements are stored in the CAS policy database, and they can be retrieved for diverse access control schemes.

### 3.4. Performance Analysis

The CAS server was modified to include supplementary user attributes in the proxy certificate, and it is expected to cause some overhead in computation and communication. We measured the time required for the creation of the \(<\text{AuthorizationDecisionStatement}>\) element in both the current CAS server and the modified CAS server to evaluate the overhead. A Java method presenting the current system time in milliseconds was adopted, and the CAS package in the Globus Toolkit 4.0.4 running on a Linux machine was used for the measurement. The machine has a 2.6 GHz Pentium 4 processor and 1 GB RAM.

Figure 8 illustrates the time for creating the \(<\text{AuthorizationDecisionStatements}>\) element in both the current and modified CAS servers. Even though there is some inconsistency, we could draw a statistically meaningful conclusion from the measurements. The mean time for the object creation in the current CAS is 8.2 msec, and that of the modified CAS is 28.9 msec. Even though the object creation time of the modified CAS is about three times of that of the current CAS, the 20 msec difference is
not a significant overhead since the CAS proxy certificate is issued only once for the user session, and the default life time of the CAS certificate is 24 hours.

Figure 8. Time for Creating <AuthorizationDecisionStatements> Element

3.5. Summary of Enhanced CAS

The Community Authorization Service (CAS) is a certificate-based access control method, and it is adopted by the Globus Toolkit for centralized security administration. However, the CAS cannot provide enough scalability, flexibility and interoperability in dynamic and heterogeneous security domains. In this chapter, we proposed an enhanced design of the Community Authorization Service (CAS). In addition to the authentication and authorization information, we enabled the CAS server to include the user attributes in the proxy certificate. The modified certificate can deliver any kind of semantic and contextual information to support diverse access control mechanisms, including Role-Based Access Control (RBAC), Metadata-Based Access Control (MBAC) [43] and
Semantic Access Control (SAC) [44]. MBAC and SAC can alleviate the scalability, flexibility and interoperability problems of the current CAS.
4. PRIVACY-PRESERVING ATTRIBUTE-BASED ACCESS CONTROL (ABAC)

Attribute-Based Access Control (ABAC) [21] is one of the flexible, interoperable and scalable access control schemes; and the scalability is increased when it is employed in an identity federation framework. The authentication and authorization processes of ABAC depend on the attributes provided by the Identity Providers. The Identity Providers in a VO authenticate its members and create attribute assertions including the required attributes of the members to help the resource (and service) providers to make authorization decisions. However, the user attributes delivered for authorization decisions are also the principal source of privacy violation.

To enhance the privacy-preserving capability of ABAC in a Grid, we developed an attribute release control mechanism to publish an optimal set of attributes that are essential to access a desired resource (or service), while exposing least amount of sensitive user information. To facilitate the selection of an optimal set of attributes, we also developed Security Policy Publication Service (SPPS) which retrieves the access condition from the access control policies in eXtensible Access Control Markup Language (XACML) [26] and converts it into a Disjunctive Normal Form (DNF) of attributes. We modified the Shibboleth Identity Provider [19] and GridShib [23, 42] for the implementation of our privacy-preserving ABAC.

4.1. Introduction

Shibboleth [5] is an attribute authorization service, and its Identity Provider publishes user attribute assertions for the authorization decision by the resource providers.
The Shibboleth Identity Provider [19] retrieves the attributes from an attribute database and put them in an assertion unless the attributes are prohibited to be published by the Attribute Release Policy (ARP) [18].

The ARP scheme helps the security system administrators protect the privacy of users by limiting the distribution of sensitive attributes, however it has limitations in supporting dynamic privacy protection in VOs. In VOs, the security requirements can be changed dynamically, and the attributes prohibited to be released by the ARP may be the key attribute for acquiring the access to certain resources (or services). Since the current ARP-based privacy protection scheme is a static mechanism, it cannot cope with the dynamic security requirement changes and cannot consider various sensitivity levels of different attributes.

We employed the GridShib [23, 42] which is a software system that interfaces Shibboleth and Grid services. The current GridShib interface for Shibboleth submits user identity to the Shibboleth Identity Provider in order to acquire an attribute assertion. The returned attribute assertion is delivered to the resource providers in order to obtain the access to their resources.

In this research, we developed a Web service, named Security Policy Publication Service (SPPS), retrieving the access condition from the access control policies in eXtensible Access Control Markup Language (XACML) [26]. The access condition can be represented as a propositional logic formula of user attributes and is returned to the clients. In order to prevent the release of unnecessary user attributes, SPPS returns only the names and data types of the attributes that are necessary to access the desired resources.
Different resource providers require different access conditions, composed of multiple attributes, for their authorization policy enforcement. This becomes an increasingly important issue as the access control policies become diverse and the number of participants in a VO increases. In order to find an optimal set of attributes to be released, the propositional logic formula representing the access condition is converted to a Disjunctive Normal Form (DNF) [11] by SPPS.

We also modified the current GridShib interface for Shibboleth to deliver the DNF of required attributes, and developed a privacy-preserving attribute release control scheme to augment the current Shibboleth Identity Provider. The modified Shibboleth Identity Provider evaluates the information sensitivity of each set of attributes satisfying the access condition, and publishes an optimal set of attributes in terms of privacy preservation.

The proposed ABAC scheme requires additional functions and mechanisms to the original security mechanism. However, its performance overhead is relatively very small because once the assertion containing user attributes is issued, the assertion is embedded in a proxy certificate which is valid for several hours (12 hours by default). The proxy certificate is used to deliver the assertion to the resource providers.

4.2. Proposed System Architecture

Figure 9 shows our privacy-preserving ABAC system designed for a Grid. We developed Security Policy Publication Service (SPPS) which retrieves the access condition involving user attributes from the access control policies in XACML and converts it into a DNF. SPPS is implemented as a Web service running on Globus Toolkit.
We modified the GridShib interface for Shibboleth to deliver the DNF of required user attributes to the Shibboleth Identity Provider. The current Shibboleth Identity Provider is augmented with a privacy-preserving attribute release control mechanism.

![Proposed System Architecture](image)

**Figure 9. Proposed System Architecture**

### 4.3. Security Policy Publication Service (SPPS)

In order for a client to gain access to a certain resource (or service) in a Grid employing ABAC and a push model, the client has to submit required user attributes to the resource (or service) provider.
The SPPS is designed to support a client to collect an optimal set of attributes for accessing an intended resource (or service). When the SPPS receives a request for the access condition for a desired resource, it authenticates the client whose distinguished name is included in the client’s proxy certificate. The resource name and intended actions are passed to the SPPS, and they are stored in the XACML request context. The SPPS searches the XACML policies to retrieve the access condition, including the names and data types of the user attributes involved. The access condition can be represented in a propositional logic formula of attributes, and the SPPS converts it into a DNF, which is a disjunction of conjunctive clauses. Each conjunctive clause in the DNF is represented as an SAML statement, and an SAML assertion containing all those statements is returned to the client. The conversion of any access condition into a DNF is based on the algorithm described in Section 5.4.

The SPPS is implemented using the APIs and Java libraries provided by the Globus Toolkit, XACML and SAML software packages. The security mechanism of the SPPS is built on the Grid Security Infrastructure (GSI) [35] provided by the Globus Toolkit.
Figure 10. SPPS Architecture

Figure 10 shows the overall architecture of the SPPS. A secure channel is established between the SPPS and a client for communication, and the Simple Object Access Protocol (SOAP) [39] is used for message passing.

The SPPS was implemented and deployed by following the five steps for building a Web service [36]:

1. Define the service interface using the Web Service Description Language (WSDL) [40].
2. Implement the service application program in Java.
3. Define the deployment parameters using the Web Service Deployment Descriptor (WSDD) [36] file and a Java Naming and Directory Interface (JNDI) [38] file.
4. Compile everything and generate a GAR archive file by using the Ant tool.
5. Deploy the service in Globus.

4.3.1. Definition of the SPPS Interface

The service interface specifies the operations that the Web service provides, the resource properties which store the service’s state information, and the message format for communication between the service and the client. Both Java and WSDL codes can be used to define the Web service. We used WSDL in order to make the SPPS language-independent.

A WSDL file contains <types>, <messages> and <portType> sections. Figure 11 shows the definition of the SPPS interface written in WSDL. The supported application names and exchanged message names are specified in the <portType> section, and the format of messages exchanged between the SPPS and the client is described in the <types> section. In the <messages> section, the names of the message types for different operations are specified.
Figure 11. SPPS Interface Definition in WSDL
4.3.2. Implementation of the SPPS

The `getAssertion()` operation within the SPPS publishes an assertion containing the names and data types of the required attributes. In order to invoke `getAssertion()`, a client submits a request message which contains the name of the desired resource, intended actions on the resource, as well as the information necessary for user authentication. The name of the desired resource and intended actions are stored in the XACML request context.

A client also can submit a service name, instead of a resource name and intended actions. When a service name is submitted by a client, the SPPS retrieves the corresponding resource name and actions from an XML file that are necessary to perform to the service.

The XACML policies are identified based on the resource names, so the SPPS retrieves the XACML policy containing the matching resource name. Figure 12 is an example of XACML request context, and Figure 13 shows a matching XACML policy. The SPPS collects the subject attributes from the matching XACML policy to provide the access condition in a DNF to the client.
<?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">
  <Subject>
    <AnySubject/>
  </Subject>
  <Resource>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" DataType="http://www.w3.org/2001/XMLSchema#anyURI">
      <AttributeValue>ftp://wsu.example.edu/example1.dat</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>

Figure 12. XACML Request Context
<?xml version="1.0" encoding="UTF-8" ?>
    <Description>...</Description>
    <Target>
        <Subjects>
            <Subject>
                <SubjectMatch MatchId="...">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">developer</AttributeValue>
                </SubjectMatch>
            </Subject>
        </Subjects>
        <Resources>
            <Resource>
                <ResourceMatch MatchId="...">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">ftp://wsu.example.edu/example1.dat</AttributeValue>
                </ResourceMatch>
            </Resource>
        </Resources>
        <Actions>
            <Action>
                <ActionMatch MatchId="...">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
                </ActionMatch>
            </Action>
        </Actions>
    </Target>
    <Rule RuleId="CommitRule" Effect="Permit">
        <Target>...</Target>
        <Condition/>
    </Rule>
</Policy>

Figure 13. XACML Policy
4.3.3. Deployment Configuration of the SPPS

The deployment process of a Web service makes it available to clients. Our SPPS is deployed in the Globus Web service container. The way how a Web service is deployed is specified by a WSDD file and a JNDI configuration as shown in Figure 14 and Figure 15, respectively. JNDI is a standard Java API for a name and directory service that that allows the clients to discover the location of Web service resources.

WSDD defines how a Web service container to disclose a Web service to the clients by presenting the URI of the Web service, operational class name that implementing the service interface, security parameters, and so on.

```
<?xml version="1.0" encoding="UTF-8"?>
<deployment name="defaultServerConfig"
xmlns="http://xml.apache.org/axis/wsdd/"
xmlns:java="http://xml.apache.org/axis/wsdd/providers/java"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <service name="SPPSService" provider="Handler" use="literal"
    style="document">
    <parameter name="className" value="org.globus.spps.services.impl.SppsServiceImpl"/>
    <wsdlFile>share/schema/spps/SppsService_service.wsdl</wsdlFile>
    <parameter name="securityDescriptor" value="etc/org_globus_spps_services/service-security-descriptor.xml"/>
    <parameter name="allowedMethods" value="*"/>
    <parameter name="handlerClass" value="org.globus.axis.providers.RPCProvider"/>
    <parameter name="scope" value="Application"/>
    <parameter name="loadOnStartup" value="true"/>
  </service>
</deployment>
```

Figure 14. WSDD File for the SPPS Deployment
4.3.4. Compile Java Programs and Organize the Files

The Java and XML files for SPPS implementation are compiled and then organized into the Globus Web service standard structure. These processes are executed by the support of Globus APIs and Apache Ant tool. As the result compilation and organization, a Grid archive (GAR) file is created. The GAR file encompasses all the classes and descriptors required for the deployment of a Web service.

4.3.5. Deploy the SPPS in Web Service Container

The final step is to deploy the SPPS in the Web service container by using a Globus API. Figure 16 shows a log file displaying the deployed Web services in the Globus Web service container.
4.3.6. Conversion of an Access Condition to a Disjunctive Normal Form (DNF)

As the required conditions for accessing resources are diverse, we need a method that can convert a complex condition composed of multiple attributes into a simple form.
in the SPPS. We can think of a resource which is accessible only by the users who can satisfy multiple conditions simultaneously. For example, a meteorology research community consisting of regionally distributed universities and laboratories has a distributed database system that stores observed data in each regional host. In order to integrate the data from a region where University-A, University-B and Laboratory-C are located, the client has to be a member of Laboratory-C and its Project-A, or should be a student working in the SchoolLab-A within the Meteorology department at University-A or University-B, or should be a faculty in the Meteorology department at University-A or University-B. This access condition can be expressed as follows, where ‘∧’ and ‘∨’ symbols stand for logical AND and OR, respectively:

\[
\text{Local-DB accessibility} = (\text{Member} \land \text{Laboratory-C} \land \text{Project-A}) \lor (\text{Student} \land \text{University-A} \land \text{SchoolLab-A} \land \text{DeptOfMeteorology}) \lor (\text{Student} \land \text{University-B} \land \text{SchoolLab-A} \land \text{DeptOfMeteorology}) \lor (\text{Faculty} \land \text{University-A} \land \text{DeptOfMeteorology}) \lor (\text{Faculty} \land \text{University-B} \land \text{DeptOfMeteorology})
\]

If a complex propositional logic formula is converted into a DNF, it is convenient to choose an optimal set of attributes which contains a least sensitive group of user attributes, while ensuring the client to access to the intended resource. We designed a simple conversion method to translate complex access conditions into a DNF as described in Section 5.4, and the conversion method is integrated into the SPPS.

### 4.4. Privacy-Preserving Shibboleth Identity Provider

The current Shibboleth Identity Provider retrieves the user attributes from the attribute repository based on the Attribute Release Policy (ARP) and inserts them in an SAML assertion to publish. However, as described in Section 2.8, the current ARP
scheme cannot provide enough privacy protection. So, we developed an attribute release control mechanism that enables the Shibboleth Identity Provider to publish an optimal set of attributes that are essential to access desired resources (or services), while exposing least amount of sensitive user information.

**4.4.1. Selection of an Optimal Set of Attributes**

In order to select an optimal set of attributes, the attribute release control mechanism considers three parameters: sensitivity of individual user attribute, trust level between interacting entities, and expected gain that can be obtained by releasing the attributes.

Thejs et al. [20] introduced a privacy assessment methodology, focusing on the privacy protection at system level. They evaluated the performance of a security system based on privacy factors such as data protection, sensitivity, transparency, and control. Among these factors, the sensitivity of data is the most important one for the selection of user attributes because others are kind of constant factors within an identity federation framework.

Leszek et al. [22] explained the relationship between trust and privacy. They claimed that the trust level between interacting entities is an important factor in analyzing the privacy protection performance of an on-line transaction system.

In our case, the risk of privacy violation caused by the release of attributes can be measured by using two factors: information sensitivity of individual user attribute and the trust level of each resource provider. Our proposed attribute release control mechanism selects a set of attributes by using a simple formula (1):

\[ \text{(1)} \]
\[
\text{Loss}_{\text{Privacy}} = \Sigma (\alpha \times \text{Sensitivity}_{\text{Attribute}} + (1 - \alpha)/\text{Trust}_{\text{Provider}})
\]

(1)

The value of \( \alpha \) in formula (1) is determined based on the weights of two factors. Expected gain that can be obtained by the release of attributes is also an important factor, and our proposed mechanism releases the selected attributes only when the expected gain, which is fixed for each resource (or service), is greater than the expected privacy loss.

4.5. Performance Analysis

We evaluated the overhead of our privacy-preserving ABAC system in terms of execution time. The majority of the overhead comes from the interaction between the SPPS server and SPPS client to obtain the access condition in a DNF of attributes. There is minor overhead caused by the attribute release control mechanism which selects an optimal set of attributes within the Shibboleth Identity Provider.

To measure the overhead, a Java method presenting the current system time in milliseconds was adopted, and the Globus Toolkit 4.2.1 running on Linux machines was used for implementation. One machine used as a local server has an Intel 2.3 GHz dual-core processor and 3 GB RAM, and the other machine used as a remote server has an Intel 2.8 GHz dual-core processor and 2 GB RAM. These two machines are connected to an Ethernet LAN. Shibboleth Identity Provider 1.3c was configured to run on the SSL-enabled Apache 2.2.8 and Tomcat 6.0.16 servers.
4.5.1. Overhead of SPPS

In our experiment, the SPPS retrieves one XACML policy out of four stored, and returns a DNF containing 13 attributes in 4 conjunctive clauses to the client. We repeated the execution 60 times and calculated the average execution times of the SPPS and the client.

Figure 17 shows the overhead caused by the SPPS in two different cases: (1) both the SPPS and the client are running on the local server, and (2) the SPPS is running on the remote server, while the client is running on the local server. In both cases, the client execution time is almost the same, but the service time of the SPPS is increased by 1,344 msec when it is running remotely, because its service time includes the network transfer time of the request and return messages.

![Graph showing overhead of SPPS](image)

Figure 17. Overhead of SPPS
4.5.2. Overhead of Privacy-Preserving Shibboleth Identity Provider

The modified Shibboleth client can submit either privacy-preserving attribute request or non-privacy-preserving attribute request to the modified Shibboleth Identity Provider. The modified Shibboleth Identity Provider verifies the request and publishes a set of attributes based on the request type. If a non-privacy-preserving request is submitted, all the attributes permitted by the ARP are published. If a privacy-preserving request is submitted, modified Shibboleth Identity Provider publishes only a set of attributes which contains the least amount of sensitive information of the user. In this case, due to the selection of an optimal set of attributes, some processing overhead is incurred.

In Figure 18, the execution time of the Shibboleth client includes the time for parsing and validating the SAML assertion returned from the Shibboleth Identity Provider. The service time of the Shibboleth Identity Provider is measured from the invocation of the attribute request until the delivery of the attributes to the Shibboleth client.

The execution time of the Shibboleth client is almost the same regardless of the request type. However, Shibboleth Identity Provider takes about additional 314 msec when the privacy-preserving attribute release control mechanism is used.
From Figures 17-18, we can see that there is processing overhead, in the order of a few seconds, when the proposed privacy-preserving ABAC is employed in a Grid. However, it is very small compared with the valid period of a proxy certificate (12 hours by default) which is used for the delivery of user attributes to the resource providers.

4.6. Summary of Privacy-Preserving ABAC System

Attribute-Based Access Control (ABAC) is a highly flexible and scalable access control scheme, however the attributes containing the sensitive information of users can be released by the Identity Providers within the identity federation framework.

In this research, we presented a privacy-preserving attribute distribution mechanism that can improve the privacy protection capability when ABAC is employed in a Grid. We employed the Shibboleth Identity Provider, the GridShib interface for Shibboleth, and Globus Toolkit for our system development.
We developed a Web service, named Security Policy Publication Service (SPPS), retrieving the access condition from the access control policies in eXtensible Access Control Markup Language (XACML) [26]. The SPPS converts a complex access condition into a Disjunctive Normal Form (DNF) of attributes, in order to facilitate the selection of required user attributes.

The Shibboleth Identity Provider is augmented with an attribute release control mechanism that can select an optimal set of attributes which minimizes the exposure of sensitive information of the user, while ensuring the access of the desired resource. The GridShib interface for Shibboleth is modified to deliver the DNF of required attributes within a SMAL assertion from the client to the Shibboleth Identity Provider.

Our privacy-preserving ABAC system incurs some processing overhead, but and it is quite cost-effective, considering the benefit of protecting the privacy of users.
5. CONVERSION OF PROPOSITIONAL LOGIC FORMULA INTO DNF BY USING A GRAPH

In this chapter, we propose an efficient method that converts propositional logic formulas into the Disjunctive Normal Form (DNF) [11, 17] using a graph structure, in order to select an optimal combination of user attributes satisfying a certain access condition.

We employed a graph structure in our DNF conversion method. Each disjunction of literals is represented by sibling nodes, and a conjunction between two literals is represented by a parent-child relationship in the graph structure. Each path from the root node to a leaf node represents a conjunction of literals in the DNF, and the DNF will be in full DNF if we remove redundant literals.

Converting a logic formula can be done by repeatedly applying the rules of logical equivalence, such as double negative elimination, the De Morgan’s law, and the distributive law. However, the complexity of the conversion process may increase exponentially as the number of literals in the logic formula increases. Our proposed conversion method can reduce the complexity of the conversion process, and it can also be used for the conversion into the Conjunctive Normal Form (CNF), with a minor modification.

5.1. Introduction

The original problem that motivated this DNF research was to find an optimal combination of user attributes that produces the least accumulated scores. Each attribute or a combination of attributes can be evaluated based on the scores assigned by their
characteristics. In order to determine the optimal combination of attributes, we need to investigate the possible combinations of attributes which satisfy the conditional requirements imposed by the problem. The conditional requirements are expressed by a logic formula which is composed of propositional literals and logical connectives, such as negation, conjunction and disjunction. Each propositional literal stands for a required attribute, and their conjunction stands for a required attribute combinations. All the logic formulas can be converted into an equivalent DNF or CNF. We propose a method for the conversion into DNF in this study because of its simplicity in the evaluation of attribute combinations. However, our proposed method can be used for the conversion into CNF, with a minor modification.

A DNF is a disjunction of logical clauses, each of which is a conjunction of literals. Each conjunctive clause may contain redundant literals and/or mutually contradictive literals. If there are same literals in a conjunctive clause, only one of them is kept. If there are mutually contradictive literals in a conjunctive clause, the clause becomes invalid, so it is removed from the DNF. After each conjunction of literals is pruned, we can check if a conjunctive clause is a subset of another conjunctive clause. If yes, the superset conjunctive clause is removed from the DNF, because we only need the smallest combination of attributes that satisfies the conditional requirements of the problem.

Each conjunctive clause is evaluated based on the property values of literals, in order to find an optimal combination of attributes with the least accumulated score. In this study, we consider only the scores of individual attributes for simplicity, however a combination of attributes may have a certain score assigned by the score assignment
policy. We consider only three logical connectives — AND (\(\land\)), OR (\(\lor\)) and NOT (\(\neg\)) — because our objective is to find an optimal combination of attributes.

The time complexity of our method is \(O(3^{n/3})\) where \(n\) is the number of literals in a given propositional logic formula.

5.2. Motivation of DNF Research

Attribute-Based Access Control [21] is widely used for Web services and Grid computing systems. The users (i.e., clients) have to submit their attributes to the service and resource providers for authorization. In order to preserve the privacy of the users, the attributes presented to the service and resource providers should be selected deliberately.

The required user attributes for services and resources can be represented in the form of a propositional logic formula, like:

\[
(\neg p \lor q ) \land r \lor (s \land t \lor u) \lor (p \land v \lor w) \land (x \land y)
\]

(2)

where \(p, q, r, s, t, u, v, w, x\) and \(y\) stand for the names of attributes. Each attribute has its property value representing the privacy level. In order to get the optimal combination of attributes that not only guarantees the authorization but also contains least privacy level, we can transform the propositional logic formula into full DNF. The DNF of the propositional logic formula (2) is:

\[
(\neg p \land r) \lor (q \land r) \lor (s \land t) \lor u \lor (p \land v \land x) \lor (p \land v \land y) \\
\lor (w \land x) \lor (w \land y)
\]

(3)
Only the user attributes are used to make an authorization decision, so each required attribute must be contained in the DNF and the optimal combination of attributes is the one with the least privacy level.

5.3. Background

The terminologies used in this chapter follow the definitions given in [12].

5.3.1. Conjunctive and Disjunctive Normal Forms

All the propositional logic formulas can be converted into a logically and semantically equivalent Conjunctive Normal Form (CNF) or Disjunctive Normal Form (DNF). Which of these two normal forms is preferred depends on the characteristics of applications [17]. The size of a CNF and a DNF can be expressed by the maximum number of literals in a conjunctive or a disjunctive clause in the CNF or DNF. Revest et al. [34] defined $k$-CNF and $k$-DNF to be the conjunctive or disjunctive normal form where each clause has $k$ literals at most.

They showed $k$-CNF is converted into $k$-DNF and $k$-DNF is converted into $k$-CNF by applying the De Morgan’s laws [34]. Both CNF and DNF are convenient formats for testing the validity of a propositional logic formula with the same complexity. Even though the complexities of DNF and CNF are the same, one may be more efficient than the other for some problems. When we test the validity of a propositional logic formula and find the optimal combination of literals, converting the propositional logic formula into DNF will be a better option, because we can test the validity of the formula and calculate the value of each combination of literals at the same time.
The propositional logic formula (3) is 3-DNF with 8 conjunctive clauses. It can be converted into 3-CNF with 8 disjunctive clauses as:

\[(p \lor \neg r) \land (\neg q \lor \neg r) \land (\neg s \lor \neg t) \land \neg u \land (\neg p \lor \neg v \lor \neg x) \land (\neg w \lor \neg x) \land (\neg p \lor \neg v \lor \neg y) \land (\neg w \lor \neg y)\]  

(4)

5.3.2. Tree vs. Graph for Representing Logic Formulas

Forms

The traditional conversion procedure that transforms a propositional logical formula into a logically equivalent CNF or DNF involves repeated use of the distributive law, associative law and De Morgan’s law. We propose a DNF conversion method that uses a graph structure, which allows fast and efficient conversion of complex propositional logic formulas into DNF.

A propositional logic formula can be represented by a tree structure for fast DNF conversion, and the cost for traversing the nodes in the tree structure is the same as that of the graph structure proposed in this study. However, the graph structure has advantages in the simplicity of programming and the size of required memory space during the computation. The tree structure requires exponential increase of memory space as the size of the propositional logic formula increases, however the graph structure requires linear increase of the memory space. If we use a tree structure, every time we meet a parenthesis in the formula, we need to build subtrees with the same number of leaf nodes. However, if we use a graph structure, we can build a subgraph.

\[(p \lor q) \land (r \lor s)\]  

(5)
Figure 19. Tree vs. Graph

Figure 19 illustrates the representation of propositional logic formula (5) using a tree structure and a graph structure. Each node, except for the ROOT node and the pseudo nodes for subtrees or subgraphs, represents a literal in the propositional logic formula. In the tree and graph, the parent and child relationship stands for a conjunctive connective, and the sibling relationship stands for a disjunctive connective. In Figure 19, we can see the advantages of using a graph structure for DNF conversion.

5.4. DNF Conversion Algorithm

There are four steps in transforming a propositional logic formula into DNF and obtain the optimal combination of attributes, which are represented by the literals.

5.4.1. Represent a Propositional Logic Formula with a Graph

Each literal in the propositional logic formula is represented by a node in a graph. Figure 20 illustrates the data structure of a node which contains the pointers to a parent...
node and three child nodes, a flag used for graph traversal, and a property value, such as the name and score of the attribute corresponding to the literal.

![Node Representation](image)

**Figure 20. Node Representation**

![Graph Representation of the Propositional Logic Formula (2)](image)

**Figure 21. Graph Representation of the Propositional Logic Formula (2)**

The graph starts with a root node which has null property value; and for each pair of parentheses, except for the first pair in the formula, a pseudo root node is created to build a subgraph of the literals within the pair of parentheses. The procedure for building a graph is as follows, starting from the root node. In the whole formula and within each
pair of parentheses, all the literals and clauses connected by OR connectives become sibling nodes, and all the literals and clauses connected by AND connectives form a hierarchy. If we construct a graph for the propositional logic formula (2), following these rules, the result is as shown in Figure 21.

5.4.2. Graph Traversal to Obtain the Conjunctive Clauses

Each path from the root node to a leaf node provides a conjunctive clause in DNF, and the property value of every visited is recorded. In Figure 21, there are two paths from the root node to a leaf node ‘x’. After the pseudo nodes represented by ‘R’ are removed from the paths, we have ‘p-v-x’ path and ‘w-x’ path, which represent two conjunctive clauses \((p \land v \land x)\) and \((w \land x)\) to be included in DNF.

If we retrieve all the paths from the root node to the leaf nodes, we can obtain the propositional logic formula (3).

5.4.3. Prune Redundancy and Remove Contradiction

The original propositional logic formula may have redundant literals and/or mutually contradictive literals. We use two levels of redundancy pruning and one level of contradiction removal.

The first level redundancy pruning and contradiction removal considers only the redundancies and contradictions with each conjunctive clause. For example, in formula (6), ‘r’ and ‘s’ appear twice, and the second ‘r’ and ‘s’ are redundant because \(r \land r = r\) and \(s \land s = s\). So, they can be removed.

\[ (p \land q \land r \land s \land p \land r \land s) \]  

(6)
There may be contradicting literals in a conjunctive clause, like:

\[(p \land q \land r \land s \land p \land \neg r \land t)\]  \hspace{1cm} (7)

In formula (7), ‘r’ and ‘\neg r’ are contradicting, which makes the whole conjunctive clause invalid. We have to check if such contradiction exists in each conjunctive clause, and if yes we need to remove the conjunctive clause from DNF because the clause never can be satisfied.

The second level redundancy pruning deals with different conjunctive clauses, instead of literals.

\[(p \land q \land r) \lor (p \land q \land r \land s \land p)\]  \hspace{1cm} (8)

In formula (8), \((p \land q \land r \land s \land p)\) is redundant because It contains additional literals than \((p \land q \land r)\); i.e., \(\{p, q, r\} \subset \{p, q, r, s, p\}\). So, it can be removed from DNF.

5.4.4. Selection of the Optimal Combination of Attributes

After the redundancy pruning and contradiction removal, we have a full DNF in which each literal appears only once in each clause. In order to select an optimal combination of attributes, we have to calculate the property value of every conjunctive clause, which is the sum of the property values of the literals within that clause. If a conjunctive clause has the best property value, then the corresponding set of attributes is the optimal combination.
5.5. Performance Issue

The complexity of our DNF conversion method is $O(3^{n/3})$ where $n$ is the number of literals in a given propositional logic formula. The most time consuming step is finding all the paths from the root node to the leaf nodes within the graph structure. The worst case is when the propositional logic formula is in a Conjunctive Normal Form (CNF) composed of a conjunction of disjunctive clauses and all of them contain the same number of literals, say $k$. In this case, the corresponding graph has $k$ nodes at every level, and the number of paths from the root node to the leaf nodes is $k^{n/k}$, which has the maximum value when $k = 3$.

5.6. Summary of DNF Conversion

DNF is a disjunction of conjunctive clauses. In order to select an optimal combination of literals that makes the propositional logic formula valid, we can convert the logic formula into a logically equivalent DNF.

In this chapter, we proposed an efficient DNF conversion method and described how to select an optimal combination of attributes that satisfies certain condition. If we consider on the validity test of a propositional logic formula, converting the formula into CNF could be a better option.

Our DNF conversion method can be easily modified to handle the CNF conversion by changing the relationships between literals. In DNF conversion, we translated each conjunctive connective into a parent-child relationship and each disjunctive connective into a sibling relationship. For CNF conversion, the translation
would be in the other way. Each conjunctive connective is translated into a sibling relationship, and each disjunctive connective is translated into a parent-child relationship. Redundancy pruning and contradiction removal can be in a similar way.
6. CONCLUSION

The purpose of this research is to develop enhanced mechanisms for the security systems in a Grid, in terms of scalability, flexibility, interoperability and privacy-preserving.

The Community Authorization Service (CAS) is a certificate-based access control method, which is adopted by the Globus Toolkit for centralized security administration. The current CAS publishes a certificate containing only the authentication and authorization information, but we modified the current CAS to issue a proxy certificate containing any kind of semantic and contextual information to support diverse access control mechanisms, including Role-Based Access Control (RBAC), Metadata-Based Access Control (MBAC), Semantic Access Control (SAC) and Attribute-Based Access Control (ABAC).

The Attribute-Based Access Control (ABAC) is a highly flexible and scalable access control scheme because it assumes distributed management of user information and a federation of access control systems in a Grid. However, ABAC has disadvantages with regard to privacy-preserving because it requires the circulation of the user attributes which can increase the risk of privacy violation.

In order to improve the privacy protection capability of the ABAC, we proposed a privacy-preserving attribute distribution mechanism for a Grid. We employed the Shibboleth Identity Provider, the GridShib interface for Shibboleth, and Globus Toolkit for our system development.
We developed a Web service, named Security Policy Publication Service (SPPS), retrieving the access condition from the access control policies in eXtensible Access Control Markup Language (XACML) [26]. The SPPS converts a complex access condition into a Disjunctive Normal Form (DNF) of attributes, in order to facilitate the selection of required user attributes.

We developed an efficient DNF conversion method using a directed graph structure. The time complexity of our method is $O(3^{n/3})$ where $n$ is the number of literals in a given propositional logic formula. It is more efficient than the traditional DNF conversion methods using the rules of logical equivalence, such as double negative elimination, the De Morgan’s law and the distributive law, and it is very easy to implement. Our DNF conversion method can be easily modified to handle the Conjunctive Normal Form (CNF) conversion with minor algorithm modification.

The current Shibboleth Identity Provider is augmented with an attribute release control mechanism that can select an optimal set of attributes which minimizes the exposure of sensitive information of the user, while ensuring the access of the desired resource.

The current GridShib interface for Shibboleth is modified to deliver the DNF of the required attributes within a SMAL assertion from the client to the Shibboleth Identity Provider.

Our privacy-preserving ABAC system incurs some processing overhead, but it is quite cost-effective, considering the benefit of protecting the privacy of users.
BIBLIOGRAPHY


