CHAPTER 5

PROPOSED SECURITY ARCHITECTURE FOR CAMPUS VIRTUAL ORGANIZATION USING ROLE BASED ACCESS CONTROL

5.1 INTRODUCTION

A number of research efforts are underway to improve the grid security of shared resources between the organizations in a VO, but the security of the users is being left under a threat since the organizations in a VO are changing dynamically, thus VO access must be established and coordinated only through trust relationships that exist between VO and the local user (Das., 2012). In general there is no trusted relationship between the classical organization and the VO or its external members. Access to the VO resources is provided to users of those organizations who possess the certificate provided by CA.

When the organizations encompassed in a VO are dynamic in nature and VO in itself is created in a dynamic manner, there arises a threat for the credentials of the user being exploited illegally. It is possible for an adversary user who has already hacked the user credential to easily misuse the resources of VO with the known credential that is revealed to the VOs. As a consequence, such systems are devoid of ensured security.

RBAC approach allows users to be assigned memberships on VO roles and role hierarchies. CAS maintains and grants users memberships on
VO roles. The resource provider in each organization has to maintain only the mapping information from VO roles to roles maintained in the local database (Anil Periera, 2006.). This approach reduces the number of entries in the role-map file. Furthermore, the resource providers can decide on the access requests of specific users by maintaining role map file for authorization information. This scheme enables the resource providers to have the complete authority over their resources. As VO is dynamic in nature, new users and resources join the VO and existing users/resources may either temporarily or permanently leave the VO. This does not affect the resource providers as CAS server takes care of grant/revoke their memberships on the VO roles. For new VOs trust relations between the CAS server and the VO may be direct or mutual, or established via intermediate Trust Management Service (Sian-Jheng., 2012).

It is seen that RBAC with CAS can produce a strong security combination. VO being a heterogeneous and distributed environment, assertion plays a crucial role which can be implemented using Security Assertion Markup Language (SAML). However, it is noted that the resource utilization of RBAC is much higher than DAC. Reducing the authentication time will further improve the performance of the VO. The following sections describe the proposed authentication mechanism.

5.1.1 Threat Analysis of Existing Certificate based Authentication Mechanism

The block diagram of threat model is shown in Figure 5.1.
A broad spectrum threat model was generated for campus grid using Microsoft’s threat modeling application for current grid security mechanisms extensively using Digital certificates (Thompson 1999, Pearlman 2002, Alfieri 2005, Buruss 2006, Lepro 2003). Figure 5.2 through Figure 5.4 shows the design and analysis of the threat model of campus grid.

**Figure 5.1 Threat model life cycle**

**Figure 5.2 Roles and service roles in the threat analysis**
External Dependencies

External dependency is a type of component which is external to the application and application does not have control over the behavior or the implementation of the dependency.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Dependency Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>External Dependency 1</td>
<td>Certificate Authority</td>
<td>P10 Service</td>
</tr>
</tbody>
</table>

Figure 5.3  The Certificate Authority dependency of Campus grid

Roles

Roles define the privilege levels users have. There are logical groups of users who use the application, and can perform the necessary functions defined by the application. These groups are then divided into sub groups, User Roles and Service Roles. User Roles contain users who interact with the application. Service Roles contain users who interact with which services are executed.

<table>
<thead>
<tr>
<th>User Roles</th>
<th>Name</th>
<th>Description</th>
<th>Auth. Mechanism</th>
<th># of Identities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Professor</td>
<td></td>
<td></td>
<td>Digital Certificate</td>
<td>1093-1096</td>
</tr>
<tr>
<td>Research Scholar</td>
<td></td>
<td></td>
<td>Digital Certificate</td>
<td>9OL-9006</td>
</tr>
<tr>
<td>Administrator</td>
<td></td>
<td></td>
<td>Digital Certificate</td>
<td>9OL-9006</td>
</tr>
</tbody>
</table>

Service Roles

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Auth. Mechanism</th>
<th># of Identities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Campus Authentication Role</td>
<td></td>
<td>Digital Certificate</td>
<td>1093-1096</td>
</tr>
<tr>
<td>VO Access Role</td>
<td></td>
<td>Digital Certificate</td>
<td>9OL-9006</td>
</tr>
</tbody>
</table>

Components

Components define the high level building blocks of the system. Components are decomposed logically into services and objects through which the user interacts with the system. Each component and object can talk to each other in order to fulfill a user action.

<table>
<thead>
<tr>
<th>Components</th>
<th>Name</th>
<th>Type</th>
<th>Tech. Type</th>
<th>Run As</th>
</tr>
</thead>
<tbody>
<tr>
<td>Campus Authentication</td>
<td></td>
<td>Website</td>
<td>Other</td>
<td>Campus Authentication Service Account</td>
</tr>
<tr>
<td>PIU Alumni</td>
<td></td>
<td>Visitor</td>
<td>Login</td>
<td>PIU Alumni Service Account</td>
</tr>
</tbody>
</table>

Threats

A threat is defined as an undesired event, a potential occurrence, often best described as an effect that might damage or compromise an asset or objective. It may or may not be malicious in nature.

Confidentiality Threats

Unauthorized disclosure of <identity> using <Campus Authentication> by <Professor>
Unauthorized disclosure of <r> using <Campus Authentication> by <Research Scholar>
Unauthorized disclosure of <c> using <Campus Authentication> by <Administrator>

Confidentiality Threat 4

Countermeasures

- Cryptanalysis Attacks: Use well-known implementations of well-known cryptographic algorithms
- Cryptanalysis Attacks: Use cryptographically generated random keys
- Cryptanalysis Attacks: Use large keys (e.g., 128-bit symmetric or 1024-bit asymmetric keys)
- Cryptanalysis Attacks: Follow principles of least privilege to provide limited access to the secret key
- Cryptanalysis Attacks: Utilize SSL or IPSec encryption to establish a secure communication channel

Integrity Threats

Illegal execution of <identity> using <Campus Authentication> by <Professor>
Illegal execution of <r> using <Campus Authentication> by <Research Scholar>

Figure 5.4  The threat report for confidentiality and integrity from the threat model
The countermeasures suggested for the campus grid include

- Using well known cryptographic algorithms
- Using cryptographically generated random keys
- Secure communication channel

Various research has implemented the recommended threat model (Thompson 1999, Pearlman 2002, Alfieri 2005, Lepro 2003) with good results in security. A major problem faced in using certificate-based authentication is based on performance, since the public key (asymmetric) cryptography is known to be at least 1000 times slower than secret key (symmetric) cryptography. Another drawback is the maintenance of the certificate revocation lists, which must be stored on public servers that have to be regularly updated.

5.2 COMMUNITY AUTHORIZATION SERVICE

Virtual communities and VO contain many members, each participating as resource provider or resource consumer or both. In such distributed environment, expressing policies in terms of direct trust relationships between producers and consumers has problems of scalability, flexibility, expressibility and lack of policy hierarchy. To resolve these problems, a trusted third party, a CAS server has been introduced that is responsible for managing the policies that govern access to a community’s resources (Pearlman et al 2002). The CAS server keeps track of membership and fine-grained access control policies of users. CAS provides a fine-grained mechanism for a VO to manage these delegated policy spaces, allowing it to express and enforce expressive, consistent policies across resources spanning multiple independent policy domains. A user wishing to access community resources contacts the CAS server, which delegates rights to the user based on the request and the user’s role within the community. The CAS architecture
builds on public key enabling a widely used set of authentication and authorization mechanisms that address single sign on, delegation and credential mapping issues that arise in VO settings.

The CAS server contains entries for Certifying Authority (CA), users, servers and resources that comprise the community and groups that organize these entities. CAS also contains fine grained policy statements and allows a separation of concerns between site policies and VO policies. It provides a mechanism for a VO to manage delegated policy spaces, allowing it to express and enforce expressive, consistent policies across resources spanning multiple independent policy domains. CAS allows VO to maintain its own set of policies. The sites then combine their local policies and enforce this combined policy. The proposed architecture implements a local CAS (L-CAS) which is managed within the VO to reduce the bandwidth overheads and at the same time use the advantages implemented in CAS for Single Sign On (SSO).

5.3 SECURITY ASSERTION MARKUP LANGUAGE

The SAML defines an XML vocabulary for sharing security assertions (Cantor et al., 2005). SAML defines an XML-based framework for describing and exchanging security information between identity providers and service providers. It is capable of providing secure single sign on solutions to internet service providers. Organization for the Advancement of Structured Information Standards (OASIS) defined SAML 2.0 standard in 2005. SAML 2.0 supports W3C XML encryption to satisfy privacy requirements. Another advantage that SAML 2.0 is the support for service provider initiated web single sign-on exchanges. This allows the service provider to query the identity provider for authentication. Additionally, SAML 2.0 adds Single Logout functionality. SAML is built on the existing web standards.
- XML
- SOAP
- XML Schema
- XML Encryption
- XML Signature
- HTTP

It defines core components that make SAML standard such as XML (Eastlake et al., 2002) based authentication and authorization assertions, request/response protocol definition, SOAP binding and profiles.

SAML assertion contains security statements which are transferred between identity provider and service provider. The security information is expressed in the form of portable SAML assertions. A SAML assertion can be unsigned, signed or signed and encrypted depending on the type of data and the sensitivity of the application requirements. The SAML standard allows message integrity by supporting X.509 digital signatures in the request/response transmissions. SAML also supports and recommends HTTP over SSL 3.0 and TLS 1.0 for situations where data confidentiality is required. There are three statements provided by the SAML assertion.

- Authentication statements
- Attribute statements
- Authorization decision statements

The use of SAML assertions with WS-Security is described in the following steps:

A SOAP message sender obtains a SAML assertion by SAML Request/Response protocol or other methods. The following steps are used to protect the SOAP message:
1. The sender constructs the SOAP message, including a SOAP header with a WS-Security header. A SAML assertion is placed within a WS-Security token. Then it is included in the security header. A key is used to construct a digital signature over data in the SOAP message body. Signature information is also included in the security header. This key is referred by the SAML assertion.

2. The message receiver verifies the digital signature.

3. The information in the SAML assertion is used for Access Control and Audit logging.

Figure 5.5 SAML assertion

SAML defines set of request / response protocols used to communicate the assertions between identity provider and service provider.

- Authentication Request Protocol
- Single Logout Protocol
• Assertion Query and Request Protocol
• Artifact Resolution Protocol
• Name Identifier Management Protocol
• Name Identifier Mapping Protocol

SAML bindings map the SAML protocol message to standard network communication protocol message format used to transport the SAML assertions between the identity provider and service provider. SAML specification provides the following bindings.

• SAML SOAP Binding
• Reverse SOAP Binding
• HTTP Redirect Binding
• HTTP POST Binding
• HTTP Artifact Binding
• SAML URI Binding

The highest SAML component level is profiles. It defines how the assertion, protocol and bindings are combined together to support use cases.

• SSO Profiles
• Artifact Resolution Profile
• Assertion Query/Request Profile
• Name Identifier Mapping Profile
• SAML Attribute Profiles

SAML contains several security mechanisms to detect and protect against such attacks. The primary mechanism relies on a PKI.
5.4 PROPOSED SECURITY ARCHITECTURE – LOW RESOURCE ACCESS CONTROL

5.4.1 Introduction

A disadvantage of using public-key for authentication is the additional computing overheads which increases the computational overheads (Bela et al.). There are many secret-key encryption methods that perform faster than most of the public-key encryption method currently available. Another disadvantage of the public key is that successful attack on a certification authority will compromise the entire security mechanism and allow an adversary to impersonate by using a public-key certificate from the compromised authority to bind a key of the adversary's choice to the name of another user. Our work Low resource – Access Control (LRAC) comprises of four parts

- Propose a novel authentication mechanism using the concept of public key being generated from the local CAS (L-CAS) rather than CA to increase speed.
- Avoid proxy credentials used in CAS by proposing a novel key and token mechanism generated dynamically and hence speed up the system.
- Assertion with CAS using SAML
- Access control based on RBAC

5.4.2 Proposed Authentication Mechanism Avoiding Proxy Credentials

In this research work an architecture is proposed which attempts to access the resources with keys and a token that are generated dynamically instead of signing in with its credentials for every access of resources by the
users. Since only request ids are passed to the VO there is no possibility of the user credentials being revealed to the VO. Also the model is designed in a way such that when a user tries to use the request id sent by another user to access the resource, it is evaluated and rejected since the request id for each user is generated dynamically for every request. Thus, the proposed architecture is simple, secure and faster than previous approaches. Table 5.1 shows the list of parameters used in the security framework. In this framework the master acts as the local CAS server.

The concept of maintaining the user credentials as well as the organizational details in a role based grid computing VO organization is explained step by step as follows:

Table 5.1 List of Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_k$</td>
<td>public key</td>
</tr>
<tr>
<td>$k_s$</td>
<td>secret key</td>
</tr>
<tr>
<td>$k_i$</td>
<td>intermediate key</td>
</tr>
<tr>
<td>$k_{vo}$</td>
<td>VO key</td>
</tr>
<tr>
<td>$C_{id}$</td>
<td>confirmation id</td>
</tr>
<tr>
<td>$R_{id}$</td>
<td>request id</td>
</tr>
<tr>
<td>$V_T$</td>
<td>validation token</td>
</tr>
<tr>
<td>$V_{id}$</td>
<td>VO id</td>
</tr>
<tr>
<td>$A_{id}$</td>
<td>aggregated id</td>
</tr>
<tr>
<td>$I_T$</td>
<td>intermediate token</td>
</tr>
</tbody>
</table>
Figure 5.6 Flow diagram of the proposed architecture

Step 1:

For every session or a particular period of time, the local CAS generates a public key $P_k$ and private key $k_s$ pair using CPAN an open source software for key generation which is maintained in the CAS.

An intermediate key $k_i$ is derived by

$$k_i = P_k^{k_s} \quad (5.1)$$

On the basis of this, a VO key named as $k_{vo}$ is calculated by multiplying $k_s$ with the arbitrary value to the power of $k_s - 1$. This is represented by the equation
\[ k_{vo} = p_k^{k_{v-1}} \# k_s \] (5.2)

CAS now sends \( k_{vo} \) to VO for further calculation that is to be performed later in this architecture. In the local CAS the keys are maintained till the end of each session.

**Step 2:**

VO stores \( k_{vo} \) as the VO key throughout the session for which the given job is being executed. It then generates a confirmation id \( C_{id} \) in response to \( k_{vo} \). VO now sends its own VO id \( V_{id} \), an arbitrary real number and its corresponding \( C_{id} \) generated randomly to CAS for validating the user, so that the CAS will hold \( V_{id} \) s for different VOs and its corresponding \( C_{id} \) s.

**Step 3:**

When a user from an organization in the VO needs to access the resources of the VO it sends a Request id \( R_{id} \) which is nothing but an arbitrary integer to CAS using SAML for assertion.

**Step 4:**

After receiving the request from the user in terms of \( R_{id} \) using SAML for assertion, the CAS can validates the received request id, \( R_{id} \) as genuine request based on SAML assertion. Once the validation is done, CAS generates a validation token \( V_T \) for the respective user which also contains the role mentioned for the user by CAS. This is done by generating an intermediate token \( I_T \) given by
\[ I_T = \left[ 1 + \frac{C_{id}}{R_{id}} \left( 1 + \frac{k_i}{p_k} \right) \frac{A_{id}}{R_{id}} \right] \]  

(5.3)

Where \( A_{id} = C_{id} + R_{id} \)

This \( I_T \) is multiplied with the \( k_s \) by the admin module of CAS and thus the final \( V_T \) is calculated. The value of \( V_T \) is provided by

\[ V_T = k_i \times I_T \]  

(5.4)

The admin sends this to the user directly since it does not want to reveal its private key to the proposed architecture module.

**Step 5:**

The user receives the \( V_T \) from the CAS and then sends it to the VO along with the \( R_{id} \) that was generated in step 4.

**Step 6:**

The Eventual Token validation process begins here. After the reception of the \( V_T \) and \( R_{id} \) from the user, the VO performs the Token validation as follows

\[ \log[V_T] - \log\left[ \frac{C_{id}}{R_{id}} k_{vo} \right] = 0 \]  

(5.5)

When the above condition is satisfied, the VO will allow the user to access the resources. Otherwise, request will be denied.
5.4.2.1 Proof for the proposed authentication model in LRAC

The entire flow of mechanism is validated by the following proof.

Let us assume that the aggregated id \( A_{id} \) is equal to the summation of Request id and Confirmation id.

\[
(R_{id} + C_{id}) = A_{id} \tag{5.6}
\]

Multiply with \( p_k \) on both sides of (5.6)

\[
(R_{id} + C_{id})p_k = A_{id}p_k \tag{5.7}
\]

Add \( C_{id}k_i \) on both sides of (6.7)

\[
(R_{id} + C_{id})p_k + C_{id}k_i = A_{id}p_k + C_{id}k_i \tag{5.8}
\]

Substitute \( k_i = p_k^{k_i} \) on both sides of (5.8)

\[
(R_{id} + C_{id})p_k + C_{id}p_k^{k_i} = A_{id}p_k + C_{id}p_k^{k_i} \tag{5.9}
\]

Divide by \( p_k \) on both sides of (6.9)

\[
R_{id} + C_{id} + \frac{C_{id}p_k^{k_i}}{p_k} = A_{id} + \frac{C_{id}p_k^{k_i}}{p_k} \tag{5.10}
\]

\[
R_{id} + C_{id} \left[ 1 + \frac{p_k^{k_i}}{p_k} \right] = A_{id} + \frac{C_{id}p_k^{k_i}}{p_k} \tag{5.11}
\]
Multiply with $k_s$ on both sides of (5.11)

$$R_{id}k_s + C_{id}k_s \left[1 + \frac{p_k^{k_s}}{p_k}\right] = k_s A_{id} + \frac{C_{id}p_k^{k_s}k_s}{p_k}$$

(5.12)

$$k_s \left[ R_{id} + C_{id} \left[1 + \frac{p_k^{k_s}}{p_k}\right] \right] - k_s A_{id} = \frac{C_{id}p_k^{k_s}k_s}{p_k}$$

(5.13)

Divide by $R_{id}$ on both sides of (6.13)

$$k_s \left[ \frac{R_{id}}{R_{id}} + C_{id} \left[1 + \frac{p_k^{k_s}}{p_k}\right] \frac{A_{id}}{R_{id}} \right] = \frac{C_{id}p_k^{k_s}k_s}{R_{id} p_k}$$

(5.14)

$$k_s \left[ \frac{R_{id}}{R_{id}} + C_{id} \left[1 + \frac{p_k^{k_s}}{p_k}\right] \frac{A_{id}}{R_{id}} \right] = \frac{C_{id} p_k^{k_s - 1} k_s}{R_{id}}$$

(5.15)

Substitute $k_{vo} = p_k^{k_s - 1} k_s$ in equation (5.15)

$$k_s \left[ 1 + \frac{C_{id}}{R_{id}} \left[1 + \frac{p_k^{k_s}}{p_k}\right] \frac{A_{id}}{R_{id}} \right] = \frac{C_{id} k_{vo}}{R_{id}}$$

(5.16)

Taking log on both sides of (6.16)

$$\log \left[ k_s \left[ 1 + \frac{C_{id}}{R_{id}} \left[1 + \frac{p_k^{k_s}}{p_k}\right] \frac{A_{id}}{R_{id}} \right] \right] - \log \left[ \frac{C_{id}}{R_{id}} k_{vo} \right] = 0$$

(5.17)

When the aforesaid condition is satisfied we conclude that the details provided by the user are valid and access to the shared resources is provided to the user.
5.4.2.2 Performance evaluation of proposed authentication mechanism

The experimental results of the proposed approach are presented in this section. The proposed approach is tested with a set of valid users and adversary users. Initially the user transmits a request id to L-CAS and L-CAS verifies this key with the confirmation key obtained from VO. L-CAS then provides Validation token which is used to validate the user. The inputs from two different user classifications are evaluated and the respective results at every stage are represented in the tabular column provided below. Some examples for valid users and invalid users are given in Table 5.2 and Table 5.3.

A user is identified to be invalid when the validation fails. Table 5.3 provided below clearly shows that the validation code has not been balanced and thus the user is evaluated as an invalid user.
<table>
<thead>
<tr>
<th>SL. No.</th>
<th>Arbitrary Value $p_k$</th>
<th>Secret Key $k_s$</th>
<th>Intermediate Key $k_i$</th>
<th>VO Key $k_v$</th>
<th>Request Id $R_0$</th>
<th>Confirmation Id $C_0$</th>
<th>Aggregated Id $A_0$</th>
<th>$\log V_T$</th>
<th>$\log \left[ \frac{C_0}{R_0} k_v \right]$</th>
<th>Validation Is User Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0</td>
<td>9.0</td>
<td>512.0</td>
<td>2304.0</td>
<td>84.0</td>
<td>34.0</td>
<td>118.0</td>
<td>6.837946</td>
<td>6.837946</td>
<td>Valid User</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>145.0</td>
<td>1.0</td>
<td>145.0</td>
<td>11.0</td>
<td>13.0</td>
<td>24.0</td>
<td>5.143788</td>
<td>5.143788</td>
<td>Valid User</td>
</tr>
<tr>
<td>3</td>
<td>7.0</td>
<td>149.0</td>
<td>8.30132762606204E125</td>
<td>1.768871116611892E127</td>
<td>53.0</td>
<td>14.0</td>
<td>67.0</td>
<td>291.66742</td>
<td>291.66742</td>
<td>Valid User</td>
</tr>
<tr>
<td>4</td>
<td>1.0</td>
<td>237.0</td>
<td>1.0</td>
<td>237.0</td>
<td>70.0</td>
<td>87.0</td>
<td>157.0</td>
<td>5.685473</td>
<td>5.685473</td>
<td>Valid User</td>
</tr>
</tbody>
</table>
Table 5.3 Evaluation for Invalid Users

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Arbitrary Value $p_i$</th>
<th>Secret Key $k_s$</th>
<th>Intermediate Key $k_i$</th>
<th>VO Key $k_{vo}$</th>
<th>Request Id $R_{id}$</th>
<th>Confirmation Id $C_{id}$</th>
<th>Validation Id $A_{id}$</th>
<th>$\log V_T$</th>
<th>$\log \left( \frac{C_{id}k_{vo}}{R_{id}} \right)$</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>1.0</td>
<td>151.0</td>
<td>1.0</td>
<td>151.0</td>
<td>12.0</td>
<td>40.0</td>
<td>52.0</td>
<td>3.8233573</td>
<td>6.2212524</td>
<td>Invalid User</td>
</tr>
<tr>
<td>2.</td>
<td>7.0</td>
<td>35.0</td>
<td>3.788186922656648E29</td>
<td>1.8940934613283236E30</td>
<td>66.0</td>
<td>87.0</td>
<td>153.0</td>
<td>69.85147</td>
<td>69.992546</td>
<td>Invalid User</td>
</tr>
</tbody>
</table>
5.4.2.3 Crypt analysis of the proposed authentication model

Crypt analysis is performed on the proposed authentication algorithm using open source crypt analysis tool. Figure 5.7 shows the design of the test bench used to analyze our proposed algorithm. Figure 5.8 shows the autocorrelation results for the given public certificate data generated in the local CAS. The entropy computed is 6.42 against 8 which is very good and compares with modern authentication algorithms.

Authentication is one part of the proposed security model which encompasses authentication, authorization and assertion. Since the resources change dynamically and policies need to be administered along multiple resources, time taken for the authentication plays a crucial part in the overall performance of the system.

Since the focus in this research is not only security but also faster access control mechanism, the proposed authentication mechanism in Low Resource Access Control (LRAC) is justified.
Figure 5.8 Autocorrelation values

Figure 5.9 Public certificate data
5.4.3 Implementation of Access Control Mechanism using SAML for Assertion and Proposed Authentication Algorithm

5.4.3.1 Introduction

In this section the implementation of the proposed authentication mechanism using SAML is described. The proposed LCAS architecture is configured to utilize the RBAC policies described in the previous section. Instead of using a CA, the proposed algorithm for authentication was implemented.

SAML by its nature was designed for carrying the security and authorization related information and have the bindings to basic transportation mechanisms. Therefore, OASIS publishes a SAML profile for the XACML (OASIS, 2005) to carry the XACML messages between the XACML actors. This profile defines the usage of SAML 2.0 to protect, transport and request XACML instances and other information. It defines six types of queries or statements;
AttributeQuery: This query may be used by the Policy Information Point (PIP)s to request attributes from Attribute Authorities or Attribute Repositories (e.g. LDAP, etc). It is a standard SAML Request. For example, this query can be used to retrieve patient’s email address to satisfy an obligation that requests the patient to be notified each time his clinical data is accessed.

AttributeStatement: The statement is the response to the attribute query giving one or more attributes.

XACMLPolicyQuery: This query is used for requesting policies from the Policy Administration Point (PAP). The element is an extension of SAML Request element. For example, this query can be used to retrieve policies specific to a patient.

XACMLPolicyStatement: This element is an extension to SAML Statement. It carries the policies requested from the Policy Administration Point (PAP).

XACMLAuthzDecisionQuery: This element is also extension for SAML Request element. It carries the XACML Request that the Policy Enforcement Point (PEP) sends to the Policy Decision Point (PDP) to request authorization decision.

5.4.3.2 Methodology

The assertion statement in SAML is as shown in Figure 5.11.

```xml
<saml:Assertion ... common info goes here ... >
  ... and here ...
  <saml:AuthnStatement
    AuthnInstant="2011-04-23T12:15:00Z"
    SessionIndex="718868823451">
    <saml:AuthnContext>
      <saml:AuthnContextClassRef>
        urn:oasis:names:tc:SAML:2.0:ac:classes:
        SecureRemotePassword
      </saml:AuthnContextClassRef>
    </saml:AuthnContext>
  </saml:AuthnStatement>
</saml:Assertion>
```

Figure 5.11 Assertion statement using custom key interface
The client side SOAP binding for attribute query is given in Figure 5.12.

Figure 5.12 Client Side Attribute Query
The server side SOAP binding for attribute and authorization decision query is given in Figure 5.13.

Figure 5.13 Server side attribute and authorization decision query
The above implementation is tested using the campus VO proposed in this work with 200 tasks being executed for each policy and with uniform communication and computation size.

5.4.3.3 Result and Discussion

The result obtained is tabulated in Table 5.4.

<table>
<thead>
<tr>
<th>VO Member</th>
<th>Time to complete task using LRAC in Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>college1</td>
<td>11.914</td>
</tr>
<tr>
<td>college2</td>
<td>12.672</td>
</tr>
<tr>
<td>college3</td>
<td>14.968</td>
</tr>
<tr>
<td>college4</td>
<td>20.050</td>
</tr>
</tbody>
</table>

From the simulation result it is clear that there is a considerable improvement using the grid environment. The comparative study is depicted as in Figure 5.14 for all categories.

![Comparisons - DAC, LRAC, RBAC](image)

Figure 5.14 Time taken by different mechanisms to compute the assigned same set of tasks
From Figure 5.14 it is seen that the performance of the VO improves with the novel mechanism proposed. The improvement in performance is linear under different policies which indicate that the proposed mechanism is scalable for very large networks.

The proposed security mechanism may be compromised internally which usually does not happen in third party PKI mechanisms. However, the grid organization manager can include third party authorization mechanisms in the LRAC server to ensure additional security. Since the focus in this research is not only security but also faster access control mechanism, the proposed authentication mechanism in Low Resource Access Control (LRAC) is justified.

5.5 THEORETICAL ANALYSIS OF APPLICATION OF GRID SECURITY MECHANISMS IN CLOUD COMPUTING

As long as Cloud computing has more homogeneous platforms and Grid computing more heterogeneous ones and dynamic resources, Grid has to consider some issues that Cloud Computing does not have: Single sign-in to access multiple Grid sites, privacy, integrity and segregation should be taken into account so that resources owned by one user cannot be accessed by unauthorized users and/or tampered with during transfer.

- Cloud users can use Cloud easily and almost instantly through a credit card, while Grid security is stricter and does not give this feature.

- The strict security approach given by Grid computing adds security, helping to prevent unauthorized access, while Cloud computing does not.
• Defining a framework which has to be used to program for the Cloud gives an additional possibility to manage security, while the more open system given by Grid does not.

• Grid computing, must share user and resource interface to allow providers to connect their resources, while Cloud Computing tries to share only the user interface while the resource interfaces are hidden.

• Since Cloud computing relies on web applications, it has the vulnerabilities of web applications, which Grid does not have.

• In Grid computing all the resources are shared with other users so the machine’s security should not be compromised. In Cloud computing, virtualization is normally done directly with the support of processors’ virtualization methods, so the resources are accessed in an abstract and more secure way.

The RBAC mechanism forms an enhanced security framework for Grids and Clouds that will allow for interoperability between technologies in the two domains. The proposed mechanisms are important because of the lack of software tools and security standards in accessing distributed HPC systems (Yilin Mo., 2012). Transporting Large Data Sets can add immensely to overheads in data processing or data integration times. The proposed LRAC model makes policy management scalable and by virtue of being modular allows more sophisticated access control models to be integrated with them.