CHAPTER 2

GRID SECURITY AND LITERATURE REVIEW

2.1 GRID COMPUTING AND DISTRIBUTED SYSTEMS

Grid computing is synonymous to high throughput with emphasis on the amount of work it can deliver over a period of time rather than the number of floating point operations it can perform in a second. Grid has evolved from existing technologies notably distributed computing, internet, Virtualization services, asymmetric cryptography and web services (Chinnici et al., 2005). The difference between Grid and Distributed systems is shown in Table 2.1.

Table 2.1 Key differentiators between distributed systems and grid

<table>
<thead>
<tr>
<th>Distributed systems</th>
<th>Grid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serve the purpose of a single organization</td>
<td>Serve the purpose of a large number of organizations</td>
</tr>
<tr>
<td>No concept of virtualization</td>
<td>Virtualization is a key concept in the grid for sharing resources</td>
</tr>
<tr>
<td>Resources are static</td>
<td>Resources are dynamic</td>
</tr>
<tr>
<td>Focus on information sharing using the client server model</td>
<td>Not limited to information sharing only, includes hardware and software sharing</td>
</tr>
<tr>
<td>Resource monitoring on a small scale and not critical</td>
<td>Grid support resource discovery and monitoring on a global scale.</td>
</tr>
<tr>
<td>Usually centralized control</td>
<td>Multiple and distributed control</td>
</tr>
</tbody>
</table>
Grid computing technology requires special software which can divide and farm out pieces of programs to thousands of computers participating in the grid (Miguel et al., 2003). It has gained substantial attention from multidisciplinary researchers worldwide due to its broad applicability and virtually unlimited resource sharing capability. It can be used for solving computationally intensive advanced science and engineering problems. Many corporations, professional bodies and universities have joined hands and are developing frameworks and software for managing grid computing projects. Large distributed computing projects using grid concepts include gpugrid.net hosted by Pompeu Fabra University where full atom molecular biology simulations are being conducted. Active research in the areas of Earth science, physics, astronomy, medicine, cryptanalysis and mathematics are being enabled using grid computing.

The resources utilized in grid computing can be high performance supercomputers, massive storage space, sensors, satellites, software applications and data belonging to multi-institutions connected through the Internet. In order to give grid environments the external appearance of a single powerful platform, open standards and middleware supporting large scale data and computation have been developed. These middleware’s hide the heterogeneity of grids (Miguel et al., 2003) and provides standardized interfaces to applications regardless the specific resources.

2.2 PUBLIC KEY INFRASTRUCTURE (PKI)

PKI is a basic building block on which network security infrastructure is built. The infrastructure contains set of software, hardware, policies and procedures needed to maintain digital certificates. It is an enabling infrastructure and technology for security mechanisms in Internet based business applications (Benantar 2001). The implementation of a PKI is intended to provide security and trust mechanisms to ensure confidentiality,
integrity, non-repudiation and authentication. The PKI framework manages keys and certificates for public-key cryptography. It consists of security services, operational policies and interoperability protocols for supporting the public-key management.

It supports Secure Socket Layer (SSL), Hyper Text Transmission Protocol over SSL (HTTPS) and Internet Protocol Security (IPSec) for network communication and transaction security. The core concept in PKI is a certificate. A certificate is a data structure containing the public key and related details about the key owner and signed by a trusted third party called Certification Authority (CA). CA is a trusted authority responsible for the management of public keys and associated certificates. The public key cryptosystem relies on the assurance of the authenticity of the public keys of the users. Thus, the role of the certificate is to bind the public keys with their owner identities.

The basic functions of PKIs are:

**Public key:** It is responsible for the generation, distribution, administration and control of keys of public-key cryptography.

**Certificate issuance:** It binds a public-key to an individual, organization or other entity.

**Certificate validation:** It verifies the validity of the certificate for a specific operation.

**Certificate revocation:** It cancels a previously issued certificate.

An important advantage of PKI is interoperability. PKI functionality is increasingly included in standard products and platforms such as Windows, Web-Services (Nadalin et al., 2004), J2EE, and .NET. The
standards related to PKIs are X.509 v3, PKIX (RFC 2459), and SPKI (RFC 2692, 93). PKI is the widely used security infrastructure for grid computing (Foster et al 1999, Thompson and Jackson 2003).

There are many reasons for PKI to be the best candidate for the grid security. PKI solves most of the security issues in grid. These are some of the advantages of PKI in grid environment.

- It provides strong authentication and one-to-many authentication mechanism when used with a certificate.
- It supports single sign on and delegation.
- It assumes no previous trust relationship between grid entities.
- It provides interoperability and scalability.
- Usability problem is significantly improved. Users are required to remember a pass phrase for his certificate only.
- It provides mobility as the certificate can be stored on a floppy disk or smart card device.
- Uniform credential through the use of X.509v3 certificates.

### 2.2.1 Public Key Cryptography

Public key cryptography supports key security properties such as confidentiality, integrity, authentication and non-repudiation. Public key encryption is based on asymmetric key cryptography. In public key cryptography a key pair is generated called public key and private key, the data encrypted by one key can only be decrypted by the other one and vice versa. The main assumption in public key cryptography is that one of the keys must remain secret (private key) and the other is made public (public key).
The public key mechanism provides confidentiality by encrypting the message. If a message sender uses a public key to encrypt a message then only the owner of the private key can decrypt the message. The basic function of public key cryptography is illustrated in Figure 2.1. It provides integrity and authentication through digital signature. RSA, Diffie-Hellman, Elliptic Curve Cryptography (ECC) and Elgamal are well known public key algorithms used in applications. The main assumption in public key cryptography is that the private key must remain secret. Thus, it needs to be adequately protected.

![Figure 2.1 Public key cryptography](image)

**2.2.2 Certificate Authorities (CA)**

A CA plays a critical role in a PKI. A CA is an institution trusted by others to guarantee for the authenticity of a public key. It acts as a trusted third party and provides various key management services. The main role of the CA is to issue digital certificates that cryptographically bind a public key to the user’s identity information. This is done by signing the information using the CA’s private key. The relying parties require the CA’s public key so that they can verify the digital signature on the certificates issued by the CA. The level of trust that a CA has depends on the level of acceptance that other
entities have in that CA. This level of acceptance depends on the policies and procedures the CA has established to ascertain the user’s identity. The CA has many other responsibilities in addition to issuing certificates. These responsibilities include generating key pairs, revoking certificates and maintaining a Certificate Revocation List (CRL). A CA can be any partner in the grid project such as a university, government or a third party. If the grid project is large enough, then it might establish its own CA.

2.2.3 Public Key Certificates - X.509

A public key certificate binds an end entity’s identity and its public key. The basic constructs of a certificate should include the name of an entity, identifying information about the entity, expiration period for the certificate and the entity’s public key. Other additional and useful information may be included in a certificate: serial numbers, the CA’s name, the CA’s public key certificate itself, the type of algorithms used to generate and verify the keys and certificate and any other information that the CA generating the certificate considers useful. The most widely used format for digital certificate is based on the Internet Engineering Task Force (IETF) X.509 standards (Tuecke et al., 2004). A detailed profile of X.509 based public key certificates can be found in the IETF RFC 2459. The X.509 version 3 is the most widely used data format for public key certificates. It provides a uniform way for expressing identity of entities on the grid. It is also standardized by the IETF. There are several types of certificates:

- Identity certificate: Contains the public key of a user and his identity together with some other information, encrypted with the secret key of the CA. This makes the certificate tamper resistant.
• Attribute certificate: Contains a set of attributes of user such as his occupation, role in a company together with some other information, digitally signed by the private key of the CA.

The extensions in the certificate are essential for the grid. GT uses X.509 end entity certificates used as the main credential for authenticating users of the grid and X.509.

### 2.2.4 Registration Authorities (RA)

A Registration Authority (RA) is a component of a PKI. The RA is the trusted representative of the CA. Most importantly RA is delegated with the CA’s explicit permission to perform tasks on behalf of the CA. The primary purpose of an RA is to verify an end entity’s identity and determine if an end entity is entitled to have a public key certificate issued. A typical function of an RA is to interrogate an end entity’s certificate request by examining the name, validity dates, applicable constraints, public key, certificate extensions and related information. Grid systems use the functionalities of the RA to authenticate VO member’s identity. The procedure of identifying a user’s identity depends on the sensitivity of the grid project and the resources involved. RA is also used to provide the CA’s public key and certificate to VO members. The VO member establishes trust with the CA by obtaining the CA’s public key and the CA’s certificate from the RA in a secure way.

### 2.3 ACCESS CONTROL MECHANISM IN GRID

After verifying the identities of users, grid security mechanisms leave access control decisions to other services. Authorization is often used as a synonym for access control. However, authorization also deals with granting access whereas access control involves deny or grant access to a resource for
a user, a group of users, a VO, a role. Various access control mechanisms have been proposed in literature with some popular mechanisms being

- Akenti Authorization Service
- Cardea
- PrivilEge and Role Management Infrastructure Standards (PERMIS)
- Shibboleth
- Virtual Organization Management Service
- Community Authorization Service (CAS)
- Role Based Access Control (RBAC)
- Attribute Based Access Control (ABAC)

2.3.1 Akenti Authorization Service

Akenti is an authorization infrastructure developed by Lawrence Berkeley National Laboratory in USA. At the most fundamental level it takes the identity of the requester and the name of the resource and returns the access rights of that user in a signed capability certificate, i.e., an authorization assertion. The Akenti model consists of resources that are accessed via a resource gateway by clients. Resource gateway act as Access control Enforcement Functions (AEF) and Akenti server acts as Access control Decision Function (ADF). The clients connect to the resource gateway using the TLS handshake protocol. At the time of the resource access, the resource gatekeeper requests a trusted Akenti server for the resource to be accessed by the client. The Akenti server finds all the relevant certificates, verifies that each one is signed by an acceptable issuer, evaluates them and returns the allowed access. Akenti requires the user to be PKI enabled and to
present an X.509 public key certificate at the time of authentication. Akenti policies are written in XML and its authorization service supports two models.

**Pull Model:** In pull model the user makes an authenticated connection to the resource, which in turn calls Akenti for authentication by passing the user and resource name.

**Push Model:** In push model, the requester calls Akenti with its name and resource name. Akenti returns a signed authorization assertion that contains the name of the requester, the resource and the access rights. The requester passes that assertion though an authenticated connection to the resource which checks if the name in the assertion matches the authenticated name and verifies the signature on the assertion.

### 2.3.1.1 Attribute and policy assertions

Akenti provides both attribute and policy assertion functions. It allows the stakeholders to create signed XML certificates containing attribute assertions and policy assertions in the form of policy certificates and use conditions. These assertions are stored and passed between the parties as signed XML formatted certificates that contain a unique certificate id and version, the signature algorithm, the certificate type, begin and end validity times and the identity of the issuer.

### Policies

Akenti supports two types of policies that are distributed and hierarchical. Akenti expects all trust relationships to be explicitly stated as part of the authorization policies. It expects all principals to be identified by
X.509 public key certificates and uses the public keys to verify all policy and attribute assertions at the time of the access decision. It contains two types of certificates: Use-Condition certificates and Policy Certificates.

To gain access to a resource, Use-Condition certificate places requirements on the attribute certificates. Use-Condition certificate contains the name of a target resource, a condition statement, a critical flag and the CA of the certificates with the attributes to be matched against these conditions and a list of rights/privileges granted. Conditions may include identity attributes that contain role or group memberships and environmental parameters. Rights are lists of actions to be performed on the targets. Action names should be unique for the whole domain of resources, irrespective of the target type. The Attribute Authority (AA) is trusted to issue attribute values that must be specified exactly, but each Use-Condition can include different AAs for each attribute. An external AA may assign attributes to users if the Use-Condition certificate lists the AA under the relevant attribute value. Akenti also supports the distributed management of attribute certificates.

A Policy Certificate states the overall policy for controlling access to a resource. It holds the trusted CAs, Stakeholders and pointers for searching Use-Condition certificates. Policy Certificates comprise a root policy certificate and optional subordinate policy certificates that inherit from the root policy. Akenti structure the target as a tree of resources. Each branch can have its own policy and inherit the policy of the superior branch. In Akenti different stakeholders can issue policies. A stakeholder is a trusted authority to issue Use-Condition certificates. Each stakeholder can impose its own access control requirements independently of other stakeholders, but one of the stakeholders can sign the Policy Certificate.
2.3.2 Cardea

Cardea (Lepro 2003) is a distributed authorization system which is developed as part of the NASA Information Power grid and implemented using the Java language. It dynamically evaluates authorization requests according to a set of relevant characteristics of the resource and requester rather than considering specific local identities. The resources to be accessed within an administrative domain are protected by local access control policies which are specified with the XACML syntax in terms of the requester and resource characteristics. Cardea leverages the XACML model for authorization evaluation and SAML for obtaining assertion data used during the evaluation process. Cardea assumes that the SAML PDP that accepts the initial request is responsible for providing the final authorization decision details to the PEP. The SAML PDP depends upon the content of the initial request to determine the correct XACML PDP to evaluate the request. Communication between components is specified directly by the XACML and SAML standards such as the request and response formats for obtaining information. Although XACML and SAML are transport independent, the initial implementation binds these protocols to the SOAP. The Cardea users are identified by X.509 proxy certificates. The exact information needed to complete an authorization decision is assessed and collected during the decision process itself. This information is assembled appropriately and presented to the PDP that returns the final authorization decision for the actual access request together with any relevant details.

2.3.3 Permis

PriviEge and Role Management Infrastructure Standards (PERMIS) is a role based authorization infrastructure funded by European Commission (EC) and developed using Java. The PERMIS (Chadwick and Otenko 2002) authorization infrastructure provides facilities for policy
management, credential management, credential validation and access control decision making. The user contacts the resource which in turn contacts PERMIS Access Control Decision Function (ADF). The PERMIS ADF makes the decision based on the user's attributes and the policy of the resource and returns this to the Access Control Enforcement Function (AEF). The AEF then enforces this decision on behalf of the resource.

In PERMIS an authorization policy is written in XML, digitally signed and secured as an X.509 attribute certificate. The policy supports hierarchical RBAC, whereby users are given roles and roles/attributes are granted access rights. Superior roles/attributes inherit the privileges of subordinate roles/attributes. The policy is stored as a digitally signed attribute certificate in a Light Weight Directory Access Protocol (LDAP) directory by its creator. User authorization tokens are attribute certificates conforming to the X.509 standard (Farrell et al., 2002, Housley et al., 2002). LDAP directory servers are used to store the attribute certificates. Attribute acquisition is normally done by a Privilege Allocator tool that creates X.509 attribute certificates and stores them in an LDAP server in the entry of the holder of the attribute certificate. Subject may possess a set of credentials from many different authentication authorities that may be pre-issued, long lived and stored in a repository or short lived and issued on demand, according to their Credential Issuing Policies.

The Subject Source of Authority (SOA) dictates which of these credentials can leave the subject domain for each target domain. PERMIS provides support for the distributed management of attribute certificates. It supports multiple externally trusted SOAs to issue roles/attributes. Thus, users can be certified internally as well as externally. The policy issuer maintains unique object identifiers (OIDs) for each policy and policies are kept in the LDAP. Each policy is validated by PERMIS engine to ensure correctness.
The name of the SOA is securely configured into the PERMIS application at start up so that there is no need to keep the policies securely.

2.3.4 Shibboleth

Shibboleth (Scott Cantor et al. 2005) is an Internet2/MACE project that provides cross-domain single sign-on and attribute-based authorization (shibboleth-arch-v05.pdf, 2002). Shibboleth has created architecture and open-source implementation for federation identity-based authentication and authorization infrastructure based on SAML (Tom Scavo et al. 2005). Federated identity allows the information about users in one security domain to other organizations. The Shibboleth System includes two major software components:

- Shibboleth Identity Provider (SIP)
- Shibboleth Service Provider (SSP).

When a user wants to access to a resource inside the same federation in a remote realm, the user has to first authenticate at his home SIP. Then home SIP sends SAML assertion containing the attributes of the user to the SSP of the resource domain. Based on these attributes, the SSP may allow or deny the user to execute the action with the remote resource. By using an attribute-based authorization model, the Shibboleth architecture is able to protect user privacy better.

2.3.4.1 Shibboleth Identity Provider (SIP)

SIP authenticates users and bind attributes in the form of SAML assertions. Each user of a federation must be registered with a SIP of that federation. SIP can be conceptually divided into four functions:
**Authentication Authority:** It issues authentication assertions to users. When a user is authenticated, the SIP provides a random identity called Shibboleth handle which is valid for short period. The reason of creating this random identity name is to keep privacy: The SIP later maps internally the random identity with the real one, but the SSP will not be able to know the user’s identity unless it is authorized by the user.

**Handle Service (HS):** The HS authenticates users in conjunction with a local organizational authentication service and issues to the user a handle token. The handle token contains SAML authentication assertion which contains user credentials. The HS is intentionally neutral to the choice of the organizational authentication mechanism and can function with almost any such service.

**Attribute Authority:** Handle token has to be produced when a user requests to access a target resource. The resource then presents the user’s handle token to the attribute authority and requests attributes regarding the user. The attribute authority enforces privacy policies on the release of these attributes, allowing the user to specify which targets can be accessed based on the attributes. The Shibboleth attribute authority retrieves attributes from an organizational authority and provides them in the form of SAML assertions.

**Single Sign-On Service:** Processes authentication requests received from SSP through the web browser to enable the single sign on. It is the function that begins the authentication process.

**2.3.4.2 Shibboleth Service Provider (SSP)**

The SSP offers protected resource access to users. The SSP decides and enforces the authorization to access resources. The SSP consumes SAML
assertions. It provides a security context for service requesters to allow or deny access to the resource. The most important functions of the SSP are:

**Assertion Consumer Service (ACS):** The ACS validates authentication assertions from the HS of an SIP. It creates a security context for the user at the SSP side.

**Attribute Requester (AR):** The AR is responsible to request attributes from the user’s SIP. The AR will exchange messages directly with the SIP’s Attribute Authority, requesting attributes that the SSP needs to decide whether the user should have access to the resource or not.

**Target Resource (TR):** The TR makes authorization decisions based on the user’s attributes.

### 2.3.5 Virtual Organization Management Service (VOMS)

VOMS (Alfieri et al 2003) is developed as part of the joint efforts of the European Datagrid and DataTAG projects. It classifies the users participating in a VO based on a set of attributes. These attributes are included into Globus-compatible proxy-certificates for supporting Single Sign On (SSO) in grid-environments.

VOMS consists of four main components:

**User Server:** receives requests from a client and returns information about the user

**User Client:** contacts the User Server with the user’s certificate, authenticates the user to the server and creates a proxy certificate with VO Fully Qualified Attribute Name (FQAN) extensions.
**Administration Client:** used by the VO administrator to add/delete and change VO-Attributes like roles and groups.

**Administration Server:** accepts requests from client to update the database. Prior to getting access to the grid the user must execute the voms-proxy-init to generate a proxy-certificate. It includes the authorization information of the user into the proxy certificate retrieved from the VOMS-Server resulting in an attribute certificate.

### 2.4Emerging Access Control Models

Two emerging models overcoming some of the disadvantages of Access Control Lists (ACL) is the

- Role based access control
- Attribute based access control

#### 2.4.1 Role Based Access Control (RBAC)

RBAC model has emerged as a powerful and generalized approach to security management and administration (Sandhu et al 2000). The basic RBAC model consists of three sets of entities called users, roles, permissions and session.

**User:** End user or Agent

**Role:** A role is a collection of permissions needed to perform a set of operations within an organization. It is related to job function or job title within the organization with associated authority and responsibility to perform operations. Each user can be assigned to one or more roles. Each role can be assigned to one or more permissions. Role can be assigned to one or more users.
**Permission:** Permission is an authorization of a particular mode of access to one or more objects in the system. Permissions can apply to objects or operations. Permission can also be assigned to one or more roles.

**Session:** Session is a mapping of one user to possibly many roles. A user establishes a session during which the user activates some subset of roles that he or she is a member. Each session is associated with a single user and each session may have different combinations of active roles.

RBAC model allows the assignment of users to roles and each role with a set of permissions. A user can acquire all the permissions of a role in which he is a member. Access control policy of a system decides on role-permission, user-role and role-role relationships. The above relationship together determines whether a particular user will be allowed to access a specific resource in the system. This model is suitable for any organization where users are assigned organizational roles with well-defined access control privileges. RBAC has the ability to modify policy to meet the changing needs of an organization. It can express a wide range of security policies including discretionary, mandatory, user-defined and organizational specific policies. Major advantage of RBAC is policy neutral and its support for security management.

RBAC is based on the principle of least privilege, separation of duties and data abstraction. It can manage any dynamic change in user’s responsibility or role within an organization by assigning a new role and revoking his assignment to any previous role.

RBAC permits permissions to be handled in terms of user job roles, thus simplifying the access control administration and presenting better manageability in enterprise environments (Ganesh Godavari and Edward Chow 2005). All users including the security administrator do not have
discretionary rights to allocate access rights. Rather than updating privileges for every user on an individual basis, the access privileges of the users essential to perform specific role are given to the users (Uday.O and Ali Pabrai 2003). RBAC has proven to be a well matched access control model to prevailing technology and business process. A number of products support some form of RBAC directly and others support closely related concepts, such as user groups that can be utilized to implement roles. Database management systems, operating systems and middleware architectures extensively use RBAC to offer access control.

2.4.2 Attribute Based Access Control (ABAC)

ABAC based access control is proposed on the basis of Web Services security specifications such as XACML, SAML and dynamic security needs of the grid computing. XACML and SAML are the two important authorization related standards that support ABAC (Eric Yuan and Jin Tong 2005). ABAC is an ideal choice when finer granularity and context-aware authorizations are required (Hai-bo Shen and Fan Hong 2006). ABAC can be used with the security standards to create portable and reusable policies which need to be enforced consistently across multiple platforms and applications. In ABAC, permissions are defined in terms of privilege-giving attributes. Instead of defining new roles to represent sets of access permissions, ABAC defines the permission sets by combining the privilege-giving attributes.

The ABAC model comprises of an Attribute Authority, Policy Enforcement Point, Policy Decision Point and Policy Authority. It defines a set of access related entities, attributes and policies. Attributes may be subject attributes, resource attributes or environment attributes.

**Subject:** An entity request service or resource. Subject may be a person or process that acts on behalf of a user.
**Resource:** A component of the system that provides or hosts grid services.

**Environment:** Environment is the context related to an invocation of a grid service.

**Attributes:** Each entity has attributes that define the identity and characteristics of the entity.

**Subject Attributes:** subject’s identity, designation, department, role, age, pincode and IP address.

**Resource Attributes:** resource’s identity, value, location and size.

**Environment Attribute:** date, time and system state.

### 2.4.2.1 ABAC based authorization framework

ABAC based authorization mechanism is designed with combined use of Web services specifications XACML and SAML. XACML provides basic authorization architecture and SAML defines a framework for exchanging security information such as authentication and authorization decisions in XML format. Hence, XACML and SAML combination is the technology for integrating existing authorization systems (Regina Hebig et al 2009).

XACML (Bray et al., 2004) defines a policy language using the attributes of requestors, resources and environment. This is the foundation for building ABAC based authorization framework. The access control framework encompasses Policy Enforcement Point (PEP), Policy Decision Point (PDP), Policy Information Point (PIP) and Policy Administration Point (PAP). The PEP intercepts the access requests from users and sends the
requests to the PDP. The PDP makes access decisions according to the security policy or policy set written by PAP and uses attributes of the subjects, the resource, and the environment obtained by querying the PIP. The PDP is the entity that implements the policy evaluation function of a policy model. It is the core of the authorization framework that makes access control decisions.

The block diagram of the ABAC authorization framework is presented in Figure 2.2.
Each policy decision is represented by PDP1, PDP2 ……PDPn. These policies essentially need their own decision functions that understand the intrinsic semantics of the policy expressions. An abstract PDP is defined which contains common characteristics of the policies. To support multiple policies, each policy is encapsulated in an independent PDP. Since the policy framework is object oriented, it is scalable and flexible, which means that new policies can be added to the framework just by inheriting the PDP class and that the existing policies can be removed and modified at any time. This framework provides PDPs, such as AccessControlList, GridMapAuthorization and other simple policy decisions. For each request policy integration point selects appropriate PDP and retrieves the attributes from the corresponding PIP to evaluate the decision function. Four types of decisions may be returned by each PDP:

- Permit
- Deny
- Not applicable
- Indeterminate

The execution sequence of the authorization framework is given as follows.

1. SOAP (Gudgin 2003) interpreter accepts incoming SOAP messages and verifies the digital signature of the message submitted within the request. It extracts the actual access request from SOAP body and forwards it to the PEP.

2. PEP forwards the request to policy integration point which invokes the corresponding PDPs and retrieves attributes by calling corresponding PIPs.
3. When the policy integration point receives the decisions returned by each PDP, it combines the decisions by using a policy combination algorithm to take a final decision and returns the decision to the PEP.

4. The PEP then executes the decision, either denying or permitting the request.

In order to integrate with the third party authorization systems, a SAMLAuthorizationCallout is established that integrates existing systems through the SAML assertions. The SAML specification defines a number of elements for making assertions and queries regarding authentication, authorization decisions, and attributes (Anderson and Lockhart 2005). As for authorization, SAML defines messages exchanged between PEP and PDP. The AuthorizationDecisionQuery element is used to send request to the PDP and an assertion returned from the PDP contains some number of AuthorizationDecisionStatements.

The SAML AuthorizationDecisionQuery element is used by SAMLAuthorizationCallout PDP to request an authorization assertion from a third party authorization service such as Shibboleth, VOMS and PERMIS.

AuthorizationDecisionQuery element includes the following elements:

- **Subject element**: It contains a NameIdentifier element specifying the identity of the requestor.

- **Resource element**: It specifies the resource for which the request to be authorized to access.
- **Action elements**: It specifies the action(s) in the request to be authorized. Generally, it may contain one or more action elements.

- **Evidence element**: It is an optional element containing one or more supporting credentials about the requestor.

The service evaluates the request against its policy and returns a response encoded in the form of a SAML Assertion. The response contains the following elements.

- **Conditions element**: It specifies the conditions in which the assertion can be used.

- **Advice element**: It specifies advice for use of the element

- **AuthorizationDecisionsStatements**: It specifies capabilities and contains the same elements as the AuthorizationDecisionQuery.

- **Signature element**: It is an optional element allowing the assertion to be verified.

The assertion returned by the third party authorization service is evaluated by SAMLAuthorizationCallout PDP. If the assertion contains a positive decision, the requested resource is permitted access or otherwise denied.

### 2.5 LITERATURE REVIEW

Longhua Zhang et al., 2003 proposed a rule based framework for role based delegation and revocation in grid security. The proposed model implements the role based delegation model 2000 (RDM2000) for delegation
and revocation of roles on a user-user delegation where a user can delegate his role to another user. The work include the identification of delegation relation, comprehensive delegation model, systematic role-based delegation policy specification using rule-based language, role delegation in role hierarchy and multistep delegation.

Geethakumari et al., 2008 proposed the role based grid delegation model (RB-GDM) approach for grid security built over the Role based access model (RBAC). Dynamic delegation, partial or restricted delegation and coarse-grained/fine-grained delegation requirements could be effectively realized using the proposed model. Different frameworks were provided for delegation requirements of intra domain and inter domain grid systems for both peer to peer relationship and hierarchical role relationships. Security may be very *coarse-grained*, meaning that all requests associated with a particular trust relationship are routed to service instance A. Security may be very *fine-grained*, meaning that a request is routed to service instance A if the requestor, acting within a trust relationship, has attributes X and Y and to service instance B if the requestor has attribute Z, or to service instance C if the request is for a operation Q.

Basit et al., 2005 proposed a policy integration framework called Integer Programming (IP) which resolves conflicts arising among heterogeneous Role based access control (RBAC) policies. Focus in this work was on violation of role assignment, violation of role specific Separation on Duty (SoD) constraint and user specific SoD constraint. The proposed method provides trade off between interoperability and preserving autonomy.

Ajith Kamath et al., 2006 proposed a policy integration technique using user credential associated with roles by improvising the existing Role based access control mechanism. Naming conflicts were sorted using schema integration technique. Conflicts in expressions were resolved using regular
expressions where complex user credentials can be described in a compact way.

Hai Jin et al., 2005 proposed RB-GACA, an RBAC based grid access control architecture specifically for grid networks which is highly scalable authorization framework. The proposed model which divides the grid into independent and dynamic domains was compared with a coarse grain access control approach using Globus tool kit. The performance evaluation using relative computation scale shows good improvements as the computation scale increases.

Chadwick et al., 2003 proposed PrivilEge and Role Management Infrastructure Standards (PERMIS) a role based access control mechanism that uses X.509 attribute certificates to store the users’ roles. In PERMIS an authorization policy is written in XML, digitally signed and secured as an X.509 attribute certificate. The policy supports hierarchical RBAC, whereby users are given roles and roles/attributes are granted access rights. Superior roles/attributes inherit the privileges of subordinate roles/attributes. The policy is stored as a digitally signed attribute certificate in a Lightweight Directory Access Protocol (LDAP) directory by its creator. User authorization tokens which are attribute certificates conforming to the X.509 standard (Farrell and Housley 2002). LDAP directory servers are used to store the attribute certificates. Attribute acquisition is normally done by a Privilege Allocator tool that creates X.509 attribute certificates and stores them in an LDAP server in the entry of the holder of the attribute certificate. Subject may possess a set of credentials from many different AAs that may be pre-issued, long lived and stored in a repository or short lived and issued on demand, according to their Credential Issuing Policies. PERMIS does not support methods to transfer attributes from attribute authority to grid services which currently is its biggest disadvantage.
Thompson et al., 2003 proposed Akenti, an access control mechanism using digital certificates which defines and enforces access policy for distributed resources. A user can assert his access requirements using use-condition certificates. Users are identified by X.509 identity certificates and hence the access control mechanism is based on policies given out by Akenti policy engine. The Akenti model consists of resources that are accessed via a resource gateway by clients. Resource gateway act as Access control Enforcement Functions (AEF) and Akenti server acts as Access control Decision Function (ADF). Akenti requires the user to be PKI enabled and to present an X.509 public key certificate at the time of authentication. Akenti has been found to be very effective for websites; however its disadvantage outweighs advantages in grid environment due to its inability to handle dynamic delegation.

Pearlman et al., 2002 proposed a community authorization service (CAS) which allows resource providers to delegate some of the authority for maintaining fine-grained access control policies to communities, while still maintaining ultimate control over their resources. AS provides the 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.

Cardea (Lepro 2003) is a distributed authorization system which is developed as part of the NASA Information Power grid. It dynamically evaluates authorization requests according to a set of relevant characteristics of the resource and requester rather than considering specific local identities. The resources to be accessed within an administrative domain are protected by local access control policies which are specified with the XACML syntax in terms of requester and resource characteristics. Cardea leverages the
XACML model for authorization evaluation and SAML for obtaining assertion data used during the evaluation process.

Long Tao et al., 2006 established a task-and-role-based access-control model for computational grid (CG-TRBAC model), integrating the concepts of role-based access control (RBAC) and task-based access control (TBAC). TBAC is specifically tailored to Workflow Management System. In a grid environment, the dependent relationship between authorization steps is usually ignored because tasks are normally isolated to each other at grid nodes. A TBAC model embodies only part of condition restrictions in the workflow context, skipping environment or system conditions that count a lot with grid security. TBAC inadequacy for grid computing is enhanced by the proposed CG TRBAC model.

Chen Ying et al., 2005 proposed Dynamic-Role Based Access Control Framework in grid Environment which introduced reputations to entities as traditional Community authorization services do not handle cheat of entities. The proposed model overcomes the problem in CAS by adopting entities dynamically according to their behavior.

Alfieri et al., 2003 proposed Virtual Organization Membership Service (VOMS) which classifies the users participating in a VO based on a set of attributes. These attributes are included into Globus-compatible proxy-certificates for supporting Single Sign On (SSO) in grid-environments. VOMS authorization is interpreted only by the resource and permissions are not provided directly.

From the above study it is observed that many of the schemes proposed in literature for access control propose only the authorization framework. However, schemes proposed for authentication and authorization rely on PKI and CA. This can lead to timing related issues in the grid.
Scalability can be achieved for large computation size, but a large number of smaller computation tasks will be affected by a combination of PKI based authentication and RBAC based authorization. Further investigations need to be carried out in this direction. It is also seen that RBAC using Akenti and VOMS are not ideally suitable for grids. Not many studies are available in the literature on the time taken for authentication and authorization for a large number of small jobs submitted in the grid.