Chapter 2

Literature Review

This chapter starts with the description of grid security requirements and the framework available/used to address these requirements. Then it presents background and related work in the areas of authentication, privacy, trust and authorization in grid systems along with the middleware available for them. The chapter also discusses the limitations of existing systems and then introduces the objectives of the thesis.

2.1 Grid Security: The Multidimensional Problem

Grid is a large scale resource sharing environment where users from different administrative domains provide and use resources among each other according to established policies. As users belong to different administrative domains and there is no centralized control, security becomes a big and challenging issue in this type of environment. Grid security is a multidimensional problem. It deals with issues like authentication, confidentiality, integrity, authorization, privacy, trust, policy *etc.* The nature, scope and applicability of these issues differ in grid systems compared to traditional client server based distributed systems. Grid systems present unique security requirements that require new technical approaches. This chapter presents a comprehensive study of grid security requirements and the approaches/mechanisms available/used to address them along with their shortcomings/limitations. Section 2.1.1 discusses grid security requirements and Section 2.1.2 discusses the security frameworks available to address these requirements. We have categorized different security requirements of grid systems under four categories namely authentication, privacy, trust and authorization. Sections 2.2 to 2.5 discuss the work done in these areas. Section 2.6 formulates the problem and Section 2.7 presents the objectives of the thesis.

2.1.1 Grid Security Requirements

The first most notable effort to identify and define unique security requirements for grid systems was made by Ian Foster *et al.* [24]. According to them, in addition to satisfy standard security requirements like authentication, access control, integrity, privacy and nonrepudiation, the security architecture must also provide support for single sign-on, delegation, protection of credentials, interoperability with local security solutions, exportability, uniform credentials/certification infrastructure, secure group communication and multiple implementations. They defined single sign-on as a feature which enables users to authenticate once and initiate computations that require resources on other sites, without further authentication. Delegation is the ability of a user to delegate some/all of his/her access rights to another entity so that the delegate can act on the original user's behalf. Interoperability with local security solutions enables local security policies to remain unchanged to accommodate inter-domain access. Exportability issue means that the security policy should not directly or indirectly require the use of bulk encryption. They also require security architecture to use a uniform credentials / certification infrastructure along with the mechanisms for protecting user credentials. The support for secure communication between dynamic groups and support for multiple implementations were also identified as important grid security requirements. The security policy must not be dictating a specific implementation technology rather it should be possible to implement the security policy with a range of security technologies, based on both shared and public key cryptography [24].

Another significant effort to describe security requirements for open grid services was started by GGF OGSA Security Workgroup [43]. According to OGSA Security Workgroup, the security challenges encountered in a grid environment can be grouped into three categories: i) Integration ii) Interoperability and iii) Trust Relationships. Following paragraphs discuss each of these challenges in brief.

Integration Challenge

Integration challenge requires security architecture to be implementation agnostic. There are numerous security frameworks, standards and implementations available today. The majority of organizations and individuals have their own preferences about the security requirements that are most suitable for their own environment. This places burden on the participants to honor the existing security frameworks and/or seamlessly integrate with them. So security architecture should be “implementation agnostic” so that it can be instantiated in terms of existing security mechanisms. Security architecture should be “extensible” also so that it can incorporate new security services when available and capable of integration with the existing security services [43].

Interoperability Challenge

Interoperability challenge deals with platform independence issues at various levels. Services that traverse multiple domains and hosting environments should be able to interact with each other, thus introducing the need for interoperability. At protocol level, different domains exchange messages across their protocol layers so they need to have interoperability at each layer of the protocol stack. Mechanisms should be there to allow domains to exchange messages in a platform neutral way. At policy level, secure interoperability requires that each party should be able to specify any policy it may wish in order to engage in a secure conversation. The policies expressed by different parties should be mutually comprehensible. At identity level, mechanisms are required to identify a user from one domain in another domain. For any cross domain invocation to succeed in a secure environment, the mapping of identities and credentials to the target domain identity is also required [43].

Trust Relationship Challenge

Trust relationship challenge deals with trust related issues among grid participants. Grid service requests can span multiple security domains. Trust relationships among these domains play an important role in the outcome of such requests. A service needs to make its access requirements available to interested clients so that they understand how to securely request access to it. The trust among the participants in a dynamic virtual organization is a complex thing to achieve and this trust must be evaluated for each session or request. This requires federation of trust relationships among the participants. The dynamic nature of the grid in some cases can make it impossible to establish trust relationships among sites prior to application execution. Given that the participating domains may have different security infrastructures, it is necessary to realize the required trust relationships through some form of federation among the security mechanisms [43].

Other efforts that also present unique grid security requirements are described in [10], [23], [44], [45], [46], [47], and [48]. According to the literature available in these papers, the security requirements for grid systems can be categorized into the following security disciplines:

i) Authentication

ii) Delegation

iii) Single Sign On

iv) Credential Lifespan and Renewal

v) Confidentiality

vi) Message Integrity

vii) Non repudiation

viii) Secure Logging

ix) Privacy

x) Trust

xi) Policy Exchange

xii) Authorization

xiii) Assurance

xiv) Manageability

Following paragraphs briefly discuss each of these security requirements:

Authentication: Authentication deals with verifying the identity of the requester. It ensures that the requester actually is the identity which he/she is claiming to be. As multiple authentication mechanisms exist, grid environment must provide integration points for them and the means for conveying the specific mechanisms utilized in any given authentication operation. The authentication mechanism may be a custom security mechanism or an industry standard technology. The authentication plug point must also be agnostic to any specific authentication technology [6], [10], [43].

Delegation: It deals with delegation of authority from one entity to another. Grid environment must provide facilities for delegation of access rights from requesters to the services. Care should be taken while delegating authority to ensure that the authority transferred through delegation is scoped only to the task(s) intended to be performed and within a limited lifetime to minimize the misuse of delegated authority [6], [10], [43].

Single Sign On: Single sign-on means enabling a user to sign on once, and then, without having to sign on again, access different domains that would normally be outside the scope of the primary sign-on domain. This capability allows a service user to utilize multiple resources with one explicit logon process, and thereafter, automatically delegate the same authenticated credential for the next resource access without user intervention, within a specific period of time. The single sign-on sessions may also include access of resources in other domains using credential delegation [6], [10], [43].

Credential Lifespan and Renewal: In grids, credentials have limited time span associated with them and most of the grid jobs take more time to execute. This may cause credentials to get invalidated and bringing the system to an invalid state. In order to avoid this, a grid system must support credential expiry notifications to users and credential revalidation facilities [6], [10], [43].

Confidentiality: Confidentiality deals with the confidentiality of the messages or information that flow over the transport mechanisms. The confidentiality requirement includes point-to-point transport as well as store-and-forward mechanisms [6], [10], [43].

Message Integrity: It provides mechanisms to detect unauthorized changes to messages. The use of message or document level integrity checking is determined by the policy, which is tied to the offered quality of service [6], [10], [43].

Nonrepudiation: Nonrepudiation guarantees that a user can't deny at a later stage that he has not performed a particular action in case he has actually performed that particular action [6], [10], [43].

Secure Logging: Secure logging is the foundation for addressing requirements for notarization, non-repudiation, and auditing. It deals with providing all services, including security services themselves, with facilities for time-stamping and securely logging any kind of operational information or event in the course of time [6], [10], [43].

Privacy: Privacy security requirement deals with protection of private information/data. It brings privacy semantics to a service usage session. Security architecture must allow both, the service requester and the service provider to specify and enforce privacy requirements and policies [6], [10], [43].

Trust: Trust security requirement deals with trust related aspects like specifying trusted credentials, trust policies and determining trustworthiness of target service. Security architecture must allow both, the service requester and the service provider to specify and enforce trust requirements and policies [6], [10], [43].

Policy Exchange: Policy exchanges allow clients and services to dynamically exchange policy information to establish a negotiated security context between them. Such policy information may contain authentication requirements, supported functionality, constraints, privacy rules *etc.* [6], [10], [43].

Authorization: Authorization is concerned with what a requester is permitted to do. It deals with issues like who can access what services and under what conditions. Authorization model should allow access to services based on established authorization policies. Authorization model should also accommodate various access control frameworks and implementations [6], [10], [43].

Assurance: Assurance provides means to qualify the security assurance level that can be expected of a hosting environment. This can be used to express the protection characteristics of the environment such as virus protection, firewall usage for Internet access, internal VPN usage, *etc.* [6], [10], [43].

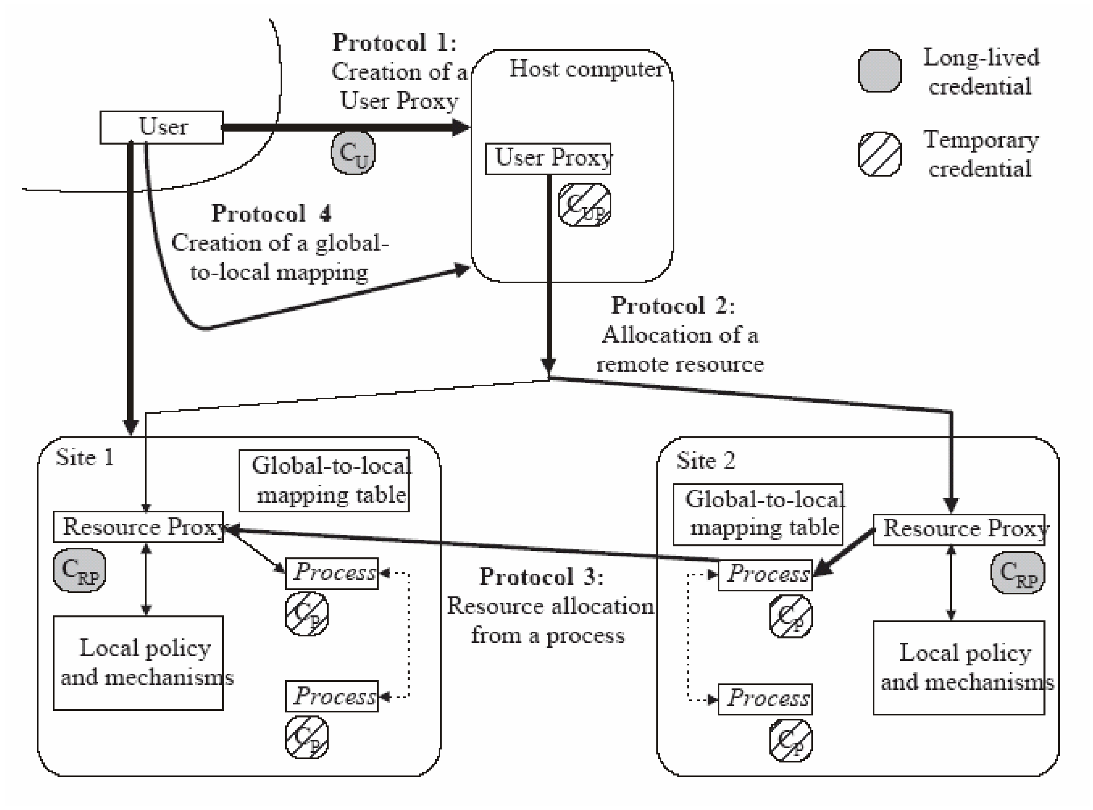
Manageability: manageability provides manageability of security functions such as identity management, policy management, security key management and other critical aspects [6], [10], [43].

In the security policy framework, the key security requirements have been categorized into four major categories related to authentication, privacy, trust and authorization. Under the authentication category, the security requirements like single sign-on, delegation, credential management, confidentiality, integrity, non-repudiation and secure communication have been included. Privacy and trust security requirements deal with privacy and trust related aspects between service providers and requesters. Authorization deals with authorization, policy specification and access control mechanisms. Assurance and manageability security requirements are not included under these categories and can be considered separately.

A number of architectures have been proposed that addresses some/all of these security requirements. Important among these architectures, along with their advantages and disadvantages, are explained in the next section.

2.1.2 Grid Security Frameworks

The security requirements described in the previous section require new technical approaches for their implementation in grid systems. The most significant security architecture for grid systems was proposed by Ian Foster *et al.* [24]. This architecture defines protocols to handle several unique grid security requirements and is the basis of GSI (Grid Security Infrastructure) in earlier versions of Globus Toolkit®. The architecture is based on security policy (a set of rules) that define the security subjects (*e.g.* users), security objects (*i.e.* resources) and relationship among them. The security policy provides a context for security architecture within which the architecture defines protocols that govern the interaction between subjects and objects. Subjects are users and processes. Objects include the wide range of resources like computers, data repositories, networks, display devices and other peripherals *etc.* [24]. Figure 2.1 shows the overview of this architecture.



Protocol 1: Creation of a User Proxy
Host computer
Long-lived credential
Temporary credential
User
User Proxy
C_U
C_{UP}
Protocol 4 Creation of a global-to-local mapping
Protocol 2: Allocation of a remote resource
Site 1
Global-to-local mapping table
Resource Proxy
C_{RP}
Process
C_P
Local policy and mechanisms
Protocol 3: Resource allocation from a process
Site 2

Figure 2.1: A Computational Grid Security Architecture [24]

The major components of the architecture are entities, credentials and protocols. The thick lines represent the protocols. The dashed lines represent the authenticated interprocess communication. The curved lines separating the user from the rest of the environment signifies that user may disconnect once the user proxy has been created.

First protocol of this architecture is user proxy creation protocol which introduces the concept of user proxy. User proxy can act on a user's behalf without requiring user intervention. It acts as a stand-in for the user. It has its own credentials, eliminating the need to have the user on-line during a computation and eliminating the need to have the user's credentials available for every security operation.

Second protocol is resource allocation protocol which introduces the concept of resource proxy. Resource proxy acts as an agent to translate between inter-domain security operations and local intra-domain mechanisms [24]. User proxy allocates resources using second protocol.

Third protocol is resource allocation from a process. While resource allocation from a user proxy is necessary to start a computation, the more common case is that resource allocation will be initiated dynamically from a process created via a previous resource allocation request. Third protocol defines the process by which this can be accomplished. Using this protocol a process created can allocate additional resources directly.

Finally the fourth protocol which is mapping registration protocol is used to define a mapping from a global to a local subject. The conversion from a global name into a local name is achieved by accessing a mapping table maintained by the resource proxy.

At time when security architecture defined in [24] was proposed, there were considerable differences between grid and web based systems. The original architecture as defined in [24] is not consistent with web services security specifications. Today, the grid services standards are converging with web services standards. A significant overlap exists between grid and web services security requirements. So the architecture proposed in [24] can be examined further for its applicability in today's grid and web services based world. The architecture also proposes to develop techniques for more flexible policy based access control mechanisms. We believe that the technologies and specifications developed for web services security can be used to realize this architecture in today's grid and web services based world. The Globus Toolkit® version 2 (GT2) is based on original architecture proposed in [24] but after that the newer versions, GT3 and GT4, of Globus Toolkit® are based on web services specifications [49]. Following paragraph briefly discusses web services security specifications, used for addressing important security related concerns of web services.

Web services security specifications tries to present a broad set of specifications that cover different security requirements like authentication, authorization, privacy, trust, integrity, confidentiality, secure communication channels, federation, policy, delegation, and auditing across a wide spectrum of application and business topologies [33]. WS-Security is the cornerstone of this effort and is submitted by IBM and Microsoft to OASIS (Organization for the Advancement of Structured Information Standards) [23]. WS-Security provides a SOAP messaging framework to integrate and support various existing security models and it proposes a set of extensions to SOAP to implement integrity and confidentiality. Although WS-Security does not in itself provide a complete security solution, it defines building blocks that web services' developers can use to build security frameworks [23]. Figure 2.2 shows web services security specifications.

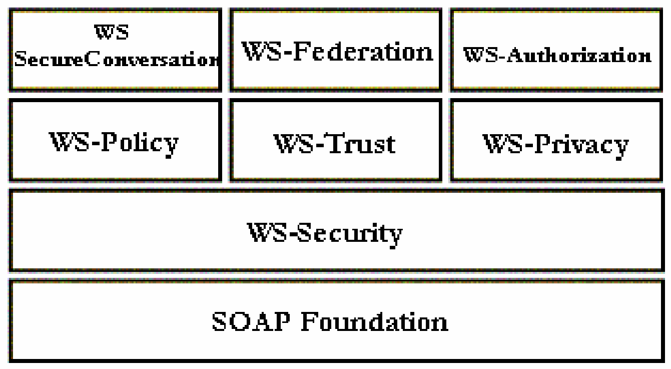


Figure 2.2: Web Services Security Specifications

This set includes a message security model (WS-Security) that provides the basis for other security specifications. These specifications address several security related issues that are also applicable to grid services. So these specifications can be used to realize the grid security model described in [24]. The specifications addresses security issues like how to associate security tokens with messages (WS-Security), how to express constraints and requirements of a web service (WS-Policy), how to request and issue security tokens to establish trust (WS-Trust and WS-Federation) *etc.* but does not give a comprehensive security architecture to address security problems present in a typical grid and web services environment. Most of other available security architectures for grid are either problem specific or not based on web services architecture. The advantage of using web services security specifications based architecture is that the tools and technologies developed for web systems can be used for grid systems also. The implemented security policy framework is making use of web services security specifications.

Another security model for grid services is proposed by GGF OGSA Security Workgroup [43]. In addition to describing unique grid security requirements, it presents a layered architecture for addressing different security related issues within Open Grid Services Architecture (OGSA). It defines a comprehensive grid security architecture that supports, integrates and unifies popular security models, mechanisms, protocols, platforms and technologies in a way that enables a variety of systems to interoperate securely [43]. This security architecture is intended to be consistent with the security model that is currently being defined for the web services framework. The various components of this security model are shown in Figure 2.3.

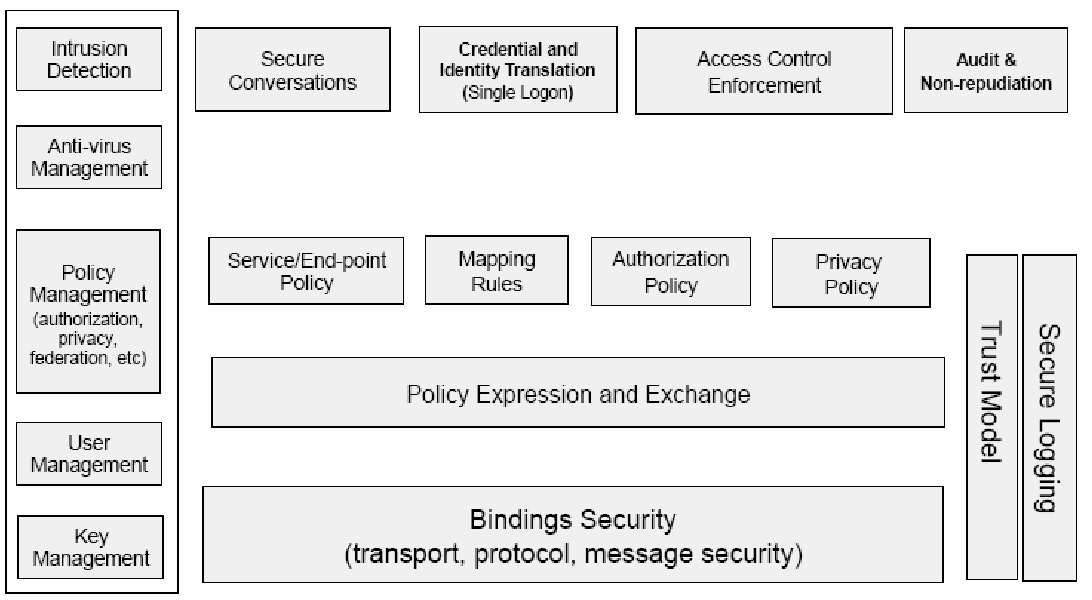

Intrusion Detection
Anti-virus Management
Policy Management (authorization, privacy, federation, etc)
User Management
Key Management
Secure Conversations
Credential and Identity Translation (Single Logon)
Access Control Enforcement
Audit & Non-repudiation
Service/End-point Policy
Mapping Rules
Authorization Policy
Privacy Policy
Policy Expression and Exchange
Bindings Security (transport, protocol, message security)
Trust Model
Secure Logging

Figure 2.3: Components of Grid Security Model [43]

The base layer of this model is binding security that deals with transport mechanisms. In this model, application specific components such as secure conversation, credential identity translation, access control enforcement, and audit and non-repudiation depends on policies and rules for authorization, privacy, identity/credential mapping, and service/end point [50]. In order to apply and manage these policies, one needs a language for policy expression and exchange and means to securely communicate through bindings to transport protocols [50]. To realize the functionality of this layer, WS-Policy or XACML specifications can be used to express and exchange policies among different entities. One side of this model has trust model that define trust anchors and how trust is derived. Other side of this model has management components that are also subject to policy enforcement [50]. This framework proposes a layered architecture but does not provide specific models to handle authentication related issues, privacy requirements, trust management and authorization and access control related issues.

There are some other efforts also to define security architecture for grid systems. Important among them include Legion [51], CRISIS [52], Global Security Architecture

[53]. Others are [54], [55], [56], [57], [58] and [59]. Some of these are based on the above systems in one or other way.

Legion is a distributed computing platform that combines large collections of independently administered machines into single, coherent environment. It is composed of independent, active objects. All entities within the system are represented by objects. Every object in a Legion system is identified by a unique, location-independent Legion Object Identifier, called LOID. One of the LOID fields stores an X.509 certificate. Legion users are given LOIDs which are registered with the system and entered in appropriate system groups and access control lists by resource providers. When an object makes a call on behalf of a user, the user's LOID and associated credentials provide the basis for authentication and authorization. In Legion, access is the ability to call a method on an object. The general model for access control is that each method call received at an object passes through a layer called "MayI" before being serviced. "MayI" decides whether to grant access according to whatever policy it implements [51]. Legion provides strong authentication and authorization capabilities but does not address privacy and trust related issues among different objects.

CRISIS [52] is a wide area security system that defines a uniform and scalable security infrastructure for wide area systems but does not address interoperability with local security mechanisms. It is also not consistent with web services specifications. The Global Security Architecture [53] presents a high level view of the overall architecture planned to be implemented in EGEE project. The security architecture inherits many of the thoughts from previous projects and parallel ongoing efforts. It is an evolutionary continuation of previous projects and efforts with new components and features added due to transition to a service oriented environment but it does not provide any specific model to address privacy and trust related issues among different entities in a grid environment. Quan Zhou *et al.* [55] propose a scalable, layered security architecture for grids but it is not consistent with web services specifications. Yuri Demchenko *et al.* [56] propose Security Architecture for Open Collaborative Environment (OCE) with the intent to build a flexible, customer-driven security infrastructure for open collaborative applications. The architecture is based on extended use of emerging web services and grid security technologies combined with concepts from the generic Authentication Authorization and Accounting (AAA) [60] and Role-based Access Control (RBAC) frameworks but this architecture also does not provide specific models to handle complex privacy and trust related issues among service providers and consumers. Similarly the models proposed by Debasish Jana *et al.* [54], Xiuying WU *et al.* [57], D. Agarwal *et al.* [58] and Syed Naqvi *et al.* [59] provide basic frameworks for handling security issues emphasizing on either one or two of the aspects from authentication, privacy, trust and authorization but none of them provide integrated support to handle authentication, privacy, trust and authorization related issues in a single framework.

A lot of other efforts are focused on defining frameworks for handling one of the issues from authentication, privacy, trust and authorization. Following sections discuss these issues and work done in these areas along with the available middleware.

2.2 Authentication

Authentication is the first and most important link in grid security chain. It deals with verifying the identity of a user. It ensures that the requester actually is the identity which he/she is claiming to be. Authentication is achieved through the presentation of some security credentials that cannot be forged [47]. In traditional client server based approaches, users are generally given some credentials that they present to the server to access the services. The server verifies the authenticity of the credentials and based on the result, it either allows or denies access to the service. Following sections briefly present the mechanisms used for authentication, problems related to authentication and the authentication solutions implemented by different middleware.

2.2.1 Authentication Schemes

The authentication problem is generally handled using three different mechanisms: based on shared secret, based on public key and based on third party. Following paragraphs discuss each of these mechanisms.

Shared secret based authentication

This mechanism works by sharing a secret. Password authentication is the most commonly used method under this scheme. A user shows its id/username and password and the server checks the id: password pair by referring to a user registry that manages a collection of such pairs. If id: password pair appears in user registry then authentication succeeds otherwise fails. One of the advantages of this mechanism is its simplicity. It is computationally inexpensive also as the password supplied by the user is checked with the hash of the password which is stored in the registry. But the limitation is that it can't be practically used to implement single sign-on and delegation requirements. So it is generally not used in grid systems.

Another way to implement the shared secret is through a challenge mechanism. Here the authenticator challenges the user to encrypt a bit of known information using the shared key. The user responds by encrypting the required information which is then validated by the authenticator. This mechanism is slightly more expensive than password authentication. The shared secret also needs to be changed periodically so that adversary cannot guess the secret.

Public key based authentication

In this type of scheme the user has a public and private key pair and the authenticator knows the public key of the user. The user encrypts the information with his private key. The authenticator can verify the authenticity by decrypting the same information with user's public key. This type of system is very secure and generally temper proof. The biggest problem in adopting the system in wide scale is the scalability of the system. If the system has thousands of users then it is very difficult for authenticator to maintain public key information of so many users. So in reality, a variation of this scheme is used which is called the certificate based system or third party authentication system [61].

Third party authentication

In this scheme the user gets a digital certificate from an entity called Certificate Authority (CA) which is known as third party. Certificates are nothing but information about the user hashed and then signed by the CA's private key. Since the public key of the CA is widely known, the authenticator has no problem in validating the certificate and hence authenticating the user to access the system based on certificate. However, this scheme requires the authenticator to trust CA. The system also requires PKI (Public Key Infrastructure) for this scheme to work. This may not be feasible always. Another mechanism of third party based authentication used in Kerberos system is to have a Key Distribution Center (KDC) which authenticates the user using a standard mechanism like using a password. The KDC generates a session key for the user to access the system encrypted with the systems public key.

These are the basic authentication schemes and their variations are commonly used in almost all types of distributed systems including wired or wireless systems [62]. In grid systems, PKI and Kerberos are the commonly used security infrastructures. PKI provides a basis to certify holders of public keys. The key constructs of PKI are the certificate and the certificate authority. A certificate is a proof of identity. A Certificate Authority (CA) is an entity that issues certificates. With a certificate, an entity can be related to its public key. Because the certificate is digitally signed, its contents can be trusted as long as the certificate issuer is trusted. The certificate format can be X.509, PGP (Pretty Good Privacy) or any other custom certificate format. In grid systems, X.509 certificates are mainly used. PKI uses asymmetric key operations. The main problem with asymmetric key operations is that they are slower than symmetric key operations. So if performance is a major concern, then symmetric key operations can also be used. In symmetric key operations each side uses same the keys based on which secure link is established to exchange secret information. In asymmetric key operations each side sends his public key and signs messages with his private key. Now each side can authenticate the other directly provided the public keys exchanged really belong to the parties involved. To ensure this, each side has its public key signed by a Certificate Authority. On the grid, the normal practice at present is for both sides to use certified public keys for authentication using X.509 certificate format.

2.2.2 Additional security requirements related to authentication

Other problems related to authentication (in grid systems) are of single sign-on, delegation and management of credentials. The execution of a grid job may span over many administrative domains. If a grid job is supposed to execute over multiple domains (on user’s behalf) after its submission to first domain, then security credentials must be passed to other domains also. For this, the user is required to sign in multiple times (once for each domain). This situation is undesirable. Support must be there for single sign-on *i.e.* the user should login just once and then be able to access the service/resources of other domains that come in the execution chain without further intervention.

Delegation is also an important requirement in grid systems. The execution of a grid job may take hours or weeks or months. So it may not be possible for users to remain online for such long times. In these situations users must be able to delegate some/all of their rights to services, so that those services can execute job on user's behalf. In grid systems, single sign-on and delegation requirements are achieved through the use of proxy certificates [63], [64].

Next problem is of management of multiple user credentials. In grid systems, the security requirements of different services vary widely with respect to the type of security credentials they need to authenticate users. Different services require different credentials like username: password, X.509 Certificates, Kerberos Tickets, Custom Security Tokens *etc.* to verify requesters. This is because services in a virtual organization are provided by service providers that belong to different physical organizations and they are free to implement any authentication mechanism of their choice to protect their services/resources. If we allow users to use different security credentials to access different services then the main problem that comes is the management of these credentials *i.e.* how and where they are to be stored, how they are to be retrieved according to the requirements of the service, how they are to be delivered to the service *etc.* So support must be there for management of multiple user credentials.

Besides these, the support to express, evaluate and enforce authentication policies should also be there. WS-Security is the most commonly used specification to describe authentication requirements of a service. It can be used to express simple authentication policies like specifying security credentials required to access a service, encryption and signature requirements of a service *etc.* But other authentication policies that include access control related information cannot be expressed easily in WS-Security. *e.g.* a policy might state that users of domain-A must have X.509 certificates and users of domain-B must have username: password in order to access a particular service. A policy might also state that a service can be accessed through username: password only on Sundays but on all other days it require X.509 certificates. The enforcement of these types of policies requires integration with access control mechanisms. So support must be there to express, evaluate and enforce authentication policies also.

Other security requirements that can be addressed along with authentication system are of nonrepudiation, integrity and confidentiality of messages. These security requirements can be handled through encryption (symmetric as well as asymmetric techniques) and digital signatures. A lot of work has already been done in these areas, so existing techniques can be used to address these security requirements.

2.2.3 Authentication in existing middleware

A number of important projects address the challenges related to authentication. Major among them include Microsoft .NET Passport system [65], Entrust TruePass portfolio [66], MyProxy [67], Kerberized Certificate Authority (KCA) [68], Password Safe [69], Microsoft Trust Bridge [70], Liberty Alliance Project [71], CredEx [72] *etc.*

The goal of Microsoft .NET Passport system [65] is to provide a uniform and distributed network to perform authentication. Passport implements single sign-on solution for the web by providing users one set of credentials (id, key, *etc.*) which are centrally stored. To use single sign-on, the target service must be Passport enabled. A user can carry a single Passport credential among several web applications that support the Passport scheme. In this manner, the user need only manage a single login name and password in order to access disparate applications. The user authenticates to Passport, Passport provides a cookie, and the user presents this cookie to the web application. Thus it functions much like Kerberos tickets [40]. It does not support proxy certificates. In our authentication framework, credentials are distributed (no central repository) over Domains (explained in chapter 5), and it supports the use of proxy certificates.

The Entrust TruePass portfolio [66] is another system that promises convenient web based authentication using PKI based Entrust Digital IDs and signatures. It also requires the target web services to be TruePass enabled. Entrust TruePass is an enhanced Web security solution that leverages the advanced public-key infrastructure (PKI) capabilities provided by the Entrust Authority™ Security Manager. It enables the protection of web site resources and applications and adds accountability and privacy to online transactions [66]. But in this credential management and delegation features are not supported.

MyProxy [67] on the other hand is a true credential repository and supports the storage of multiple user credentials. It is developed to meet the credential management requirements of the grid community. The MyProxy system is the implementation of the Virtual Soft Token system proposed in [73], where the X.509 proxy certificates are used to store and retrieve user credentials without having to expose the private key. MyProxy is being extended with a name change to GridLogon to reflect its enhanced capabilities.

The main function of the GridLogon service is to issue X.509 credentials to clients that are used by the client to securely access Globus Toolkit® enabled Grid resources [74].

KX.509 1.0, from the University of Michigan, is a Kerberized client-side program that acquires an X.509 certificate, using existing Kerberos tickets. The Globus Toolkit® [75] normally uses X.509 credentials [76], [77]. KX.509 allows a user to authenticate to a host that is running Globus software using Kerberos tickets instead of requiring X.509 certificates to be installed [68]. X.509/KCA is a system that just supports the exchange of Kerberos ticket-granting ticket for X.509 proxy certificate but it is not a credential repository.

Another system Password Safe [69] is an example of a password database that allows users to store passwords for different accounts. Password Safe protects passwords with the Blowfish encryption algorithm. The problem with Password Safe is that it involves the exchange of a single type of token only. It is not a single sign-on and delegation solution.

The Liberty Alliance Project [71] is an effort, initiated by Sun Microsystems and now involving many companies, with the goal of facilitating authentication on the web for both human users and automated mechanisms. Liberty Alliance Project defines how various identity providers (which store information related to user identities) can be linked together to support single sign-on across multiple distinct services. The framework provides standards for sharing attribute information across a trusted set of entities called the Circle of Trust (COT). The user is authenticated by the identity provider and after authentication, user can access services provided by service providers. The sessions are transferred between different service providers. The user may possess different identities with different service providers as long as they are connected through the circle of trust constraint. The weakness of Liberty Alliance Project is that it falls short of addressing support for diverse authentication token types [72].

Microsoft's TrustBridge [70] is federated identity effort that builds on WS-* set of web services security specifications like WS-Security, WS-Trust, WS-Federation *etc.* TrustBridge enable businesses to share user identity information between applications and organizations. It allows different organizations using the Windows operating system to exchange user identities and interoperate in heterogeneous environments using industry-standard XML web services protocols.

CredEx [72] is an open source standards based web service that facilitates the secure storage of credentials and enables the dynamic exchange of different credential types using WS-Trust token exchange protocol. With CredEx, a user can achieve single sign-on by acquiring a single credential but support for credential delegation is not there. MyProxy [67] and CredEx [72] are indeed a key inspiration for the authentication framework.

More attempts in this area also exist like [78], [79], [80], [81] and [82] but these do not support all types of authentication features required in grid systems. The implemented authentication framework is distinguishable from the discussed authentication systems in different ways *e.g.* the implemented framework is based on web services security specifications like WS-Security, WS-SecureConversation *etc.* It supports the storage of multiple types of tokens/credentials. (*e.g.* X.509, Kerberos, Custom Security Token *etc.*). It can be used for grid as well as web services. It provides a credential manager service which is distributed over different domains so removes the drawbacks of central storage. It provides support for single sign-on and delegation through proxy certificates. It provides a mechanism to express, evaluate and enforce authentication policies also. These are some of the characteristics of the implemented authentication framework that make it different from other systems.

2.3 Privacy

Privacy is becoming a serious concern in service oriented environment like grid and web, where a large number of users provide their general as well as personal information to service providers to gain access to their services. From service requester's point of view, privacy deals with ensuring that service providers are not misusing service requester's data and are using the data only for the purpose for which it has been submitted. Service providers generally publish their privacy policies in plain English language to ensure users that information provided by them will not be used inadequately. But the claims made by them (which are written in plain English) can't give users the guarantee that their information will not be misused [83]. The privacy model implemented by service providers must have the ability whereby uses have control to check the misuse of their data as well as control over how their information can be collected, stored, used and shared.

2.3.1 Privacy Issues

A Privacy Model should address a wider notion of privacy that focuses on providing users with the purpose of data collection, choice regarding collection and conditions under which data can be used so that users can control and make decisions regarding the disclosure of their private information. It must support privacy requirements like individual control, collection limitation, purpose specification and consent *etc.* [84].

In addition to the assurance of proper privacy of personal data, users generally expect more privacy features from service providers [85]. Sometimes users want to hide their actual identity while accessing a service. *e.g.* consider an online system that provides e-voting services. Here individuals may wish to hide their identity while voting for a particular party but not the fact that they are voting/accessing a service. Sometimes users want to hide even from the service provider that they have accessed a particular service. *e.g.* a user may want to hide from service provider that he has accessed a service to purchase a video titled "xyz", because, if service provider know that a particular user is accessing a particular service with particular parameters then he can observe user's usage pattern and make interpretations (data mining). So some support should be there to handle these types of privacy requirements. Though this issue can be resolved by granting anonymous access to services but this does not give a complete solution because if service providers do not know about user preferences then personalized access to services can not be provided. This is likely to disadvantage users as they will not be able to customize services according to their interests and hence will not experience quality of service. In fact, participation in the real world requires disclosure of information to some extent [86]. To get customized access to services according to our preferences, it is necessary to supply preference details (private information) to service providers.

Most of the work in privacy is oriented towards handling the privacy requirements of service requesters. Besides handling privacy requirements of service requesters, privacy concerns of service providers should also be addressed. *e.g.* consider a secret service is provided by government to some of the officials of a department. If the officials have access to this secret service then they can leak information regarding security credentials required to access the service, address of the service, input and output from the service *etc.* Clearly, government in this case requires that officials must not know the credentials/attributes required to access the service, the actual address of the service and the input and output requirements of the service. Most of the privacy models address privacy requirements of service requesters and not of service providers. But the scenario just described demands a privacy model which must be able to handle privacy requirements of service providers also.

The access control mechanisms implemented by authorization framework also need to be extended to include privacy based access to data and services. The access control decisions should not be determined by the user's identity alone, but by the task the individual user is currently performing and his conformance to established privacy policies. Personal data should be accessible to a user only if such an access is necessary to perform his task and if he is authorized to perform this task [87].

The privacy model should also allow service providers to express, evaluate and enforce privacy policies on their resources/services to protect them. Web services security specifications propose the use of WS-Privacy to address privacy related issues between service providers and requesters but it has not been completed as of June 2008. WS-Privacy will use a combination of WS-Policy, WS-Security and WS-Trust to communicate privacy policies. The privacy policies are stated by organizations that are deploying web services and require that incoming SOAP requests contain claims that the sender conforms to these privacy policies. WS-Security specification can be used to encapsulate these claims into security tokens, which can be verified. To express access control privacy policies, XACML can also be used. XACML, as already discussed, is eXtensible Access Control Markup Language that is used to express general access control related policies. As it is extensible, it can be used to express a variety of other security policies also. In the implemented privacy framework, we are making extended use of XACML to express privacy related access control policies.

2.3.2 Privacy handling in existing middleware

Most of the software available to construct grid and web services do not provide effective methods to handle complex privacy requirements of both the service providers and the service requesters. Much of the work done in the area of privacy in distributed systems like grid is limited. There are some projects that address privacy requirements of grid systems in one or other way *e.g.* Shibboleth [88], PRIMA [89], PERMIS [90], GT4 [76] but they are mainly policy based authorization frameworks. The efforts like P3P [91] and EPAL [92] are primarily focused on addressing privacy related issues in web systems.

Shibboleth [88] is an attempt to address privacy in an authorization environment but it is primarily focused on using pseudonymity. It does not provide a complete privacy based access control environment. It is a tool for identity federation between campuses that allows resources to obtain attributes about the user (*e.g.* departmental affiliation, student status), while preserving the user's privacy and not having to become involved with the details of how the user is authenticated in their home domain [74].

The efforts like PRIMA [89] and PERMIS [90] mainly provide policy based authorization framework but provide little support to address complex privacy related issues. They do not provide purpose based access to grid services. The most commonly used toolkit to construct grid services GT4 [76] also does not provide any comprehensive privacy model. The support is also missing from Alchemi [93] which is used to construct grid applications in .NET environment.

Another effort, P3P [91], is a XML document that describes the data collection practices for a site. It is a specification by the World Wide Web Consortium (W3C) and provides a framework for informed online interactions on the web, allowing users to negotiate agreements with services that declare their privacy practices and request data. P3P provides a vocabulary and standard machine readable format to allow websites to declare their privacy practices and describe the data they collect. The user privacy preferences are described using a P3P Preference Exchange Language (APPEL 1.0 [94]). Users can use this language to express their preferences as a set of rules (called a ruleset) which can then be used by their agent to make automated or semi-automated decisions regarding the acceptability of machine-readable policies from P3P compliant web sites [84]. It is mainly used in web based systems to publish privacy policy of an organization to requesters but it does not provide any technical mechanism to check non conformance of these policies at service provider’s end. P3P specification complements the implemented privacy model in the sense that it can be used for declaring and negotiating privacy policies.

EPAL [92] is an XML-based markup language that formalizes enterprise-internal privacy policies. It approaches the problem on the server side and addresses the need of a company to specify access control policies, with reference to attributes/properties of the requestor, to protect private information of its users. EPAL is designed to enable organizations to translate their privacy policies into IT control statements and to enforce policies that may be declared and communicated in P3P [84]. EPAL does not provide any comprehensive privacy model to support anonymous access, hidden services access and purpose based access. Moreover it does provide any procedure to integrate privacy based access control mechanisms with authorization framework.

Marc Langheinrich [95] suggests a P3P-style architecture to provide notice of data collection in ubiquitous computing environments. In this, sensors and other recording devices use a P3P declaration format to announce their data collection practices. User agents on user mobile devices release contextual information based on a comparison between the privacy declaration and user preferences encoded in a machine readable format. The proposed Privacy Awareness System (paws) implements P3P in ubiquitous computing environments to provide notice-choice based data collection.

Simone Fischer-Hubner *et al.* [87] augmented a task-based access control model with the notion of purpose and consent. Here data can be accessed in a controlled manner only by executing a task. A requester can access personal data if this access is necessary to perform its current task and he is authorized to execute this task. Additionally, the task's purpose must correspond to the purposes for which the personal data was obtained or there has to be consent by the data subjects. A task consists of a set of certified operations representing the set of "necessary accesses" to perform that task. However, the model does not consider context-dependent access control or obligations. It is also not based on web services security specifications or standards.

P. Bonatti *et al.* [96] developed a language for use-based restrictions that allows one to state under which conditions specific data can be accessed. In this language, a data user is characterized as the triple user, project, and purpose. Projects are named activities registered at the server, for which different users can be subscribed, and which may have one or more purposes. Each user and project is associated with a profile, which captures properties such as name and address or title and sponsor. However, the language does not support obligations and consent. Gunter Karjoth *et al.* [83] propose a privacy model for enterprises. The model is not based on open standards but is indeed a key inspiration for our privacy model.

Other approaches used to handle privacy requirements and policies are described in [84], [97], [98], [99], [100] and [101]. Ajay Brar *et al.* [84] describes at a higher level the framework for providing privacy-aware personalization in ubiquitous computing environments but do not present any comprehensive privacy model and its integration with authorization framework. Scott Lederer *et al.* [97] present a conceptual model for everyday privacy in ubiquitous computing environments. Stephen Crane [98] uses the concept of trust agents for preserving privacy. Both, [97] and [98] are not based on standards and do not support purpose based access. U.M. Mbanaso *et al.* [99] propose a Trust Authorization Framework (TAF) that builds on the capabilities of XACML to support the bilateral exchange of policies and credentials through trust negotiation. The framework has the capabilities to protect resources, policies and credentials simultaneously in distributed environment for users with or without pre-existing trust relationships but does not support purpose based access. Wei Xu *et al.* [100] addresses consumer privacy concerns in the context of highly customizable composite web services. The approach involves service producers exchanging their terms-of-use with consumers in the form of “models”. The framework also provides automated techniques for checking these “models” at the consumer site for compliance of consumer privacy policies but the framework does not make use any standard (*e.g.* XACML) for specifying privacy policies. It uses its own mechanism for specifying privacy policies which is based on the concept of “labels”. Christian Schlager *et al.* [101] introduced the idea to use attribute based access control mechanisms with XACML policies for building a flexible and privacy aware system. They have tailored the model for b2c eCommerce applications but the model does not support purpose based access.

To our knowledge, no software is available to handle complex privacy requirements between service providers and service requesters while developing grid and web services. Many of them are not based on open standards and use their own mechanisms for handling privacy requirements and to express, evaluate and enforce privacy policies. In the implemented privacy model, we have made an attempt to define a general, standards based privacy model that is able to address the privacy issues of both, the service requesters and service providers. The model supports anonymous access and hidden services access through the concept of anonymous and custom security tokens. The model also provides mechanisms to express, evaluate and enforce privacy policies. It also describes the integration of privacy based access with authorization framework.

2.4 Trust

The context of grid computing introduces challenging trust related issues as both, resource providers and resource consumers can come from mutually distrusted administrative domains and any of them can behave maliciously. The usage of grid system can become severely limited if participants are not given any means to access the trustworthiness of each other in the environment. To achieve faithful domain to domain interaction and confidence of participants in accessing/providing resources and services from/to other administrative domains, it is extremely important for domains to address trust issues by introducing a trust model and integrating that model into authorization framework to enable trust based access to resources and services.

Trust is a fascinating subject and social scientists have researched into the concept and developed theories around it. It can be defined in different ways at different levels. It is a subjective and elusive notion. The concept of trust is excessively complex and appears to have many different meanings depending on how it is used [102]. Trust is a multifaceted issue that may be related to other attributes such as reliability, accuracy, honesty, risk, competence, security, belief, perception, utility, benefit, expertise *etc.* [102], [103]. Trust is generally defined as having confidence that a service/resource will behave in an expected manner despite the lack of ability to monitor or control the environment in which service/resource operate [47]. Diego Gambetta [104] has defined trust as *"a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action"*. Grandison and Sloman [105] define trust as *"the firm belief in the competence of an entity to act dependably, securely, and reliably within a specified context"*. Another good definition of trust is given in [106] as: *"Trust is the firm belief in the competence of an entity to act as expected such that this firm belief is not a fixed value associated with the entity but rather it is subject to the entity's behavior and applies only within a specific context at a given time"*. This definition captures many important properties of trust as: i) Trust is not a fixed value, it is dynamic. ii) It is context and time dependent. iii) It depends on entity's past and present behavior. Trust can also be affected by those actions that we cannot monitor. Another property of trust is that it is not transitive. *i.e.* if A trusts B and B trusts C then it does not mean that A trusts C. Transitivity may hold if certain conditions are met but in general, trust is not always transitive [107].

2.4.1 Taxonomy of Trust

This section describes important terms and concepts related to trust like trustworthiness, reputation, feedback, opinion, trust functions, their categories, types of trust *etc.* as defined in [48] and [108].

Trustworthiness

An entity's trustworthiness is an indicator of the quality of the entity's services. It is often used to predict the future behavior of the entity. If an entity is trustworthy, it is likely that the entity will provide good services in future transactions also. Trustworthiness is generally given a value from 0 to 1. However, trustworthiness can be a continuous function also.

Reputation

Reputation indicates the general perception of a system's trustworthiness as perceived by other entities. The concepts of reputation and trust are closely related. Reputation metric is generally used by individuals based on word-of-mouth, or past history to determine the trustworthiness of a person or other things. Alfarez Abdul-Rehman *et al.* [107] define reputation as *"an expectation about an individual's behavior based on information about or observations of its past behavior."* In online communities, where an individual may have much less information to determine the trustworthiness of others, their reputation information is typically used to determine the extent to which they can be trusted. Similarly, in a distributed grid environment, where the grid nodes join or leave the grid, the reputation information can be used to determine the trustworthiness of the nodes.

Feedback

Feedback is a statement given by the user about the quality of a service provided by a service provider. Feedback can be multidimensional, reflecting the user's evaluation on a variety of aspects of a service, *e.g.*, price, product quality *etc.* But for simplicity, it is generally taken to be one-dimensional.

Opinion

An opinion is a user's general impression about a service provider. It is derived from the feedback on all the transactions that are conducted with the service provider. Similar to feedback, opinion also can be multidimensional in nature.

Trust Function

A trust function is a set of metrics used to infer the trustworthiness of an entity. Following are the different categories of trust functions as defined in [108].

Subjective vs. Objective

An entity's trustworthiness is often related to the quality of services it provides to others. If the quality of a service can be objectively measured, then an entity's trustworthiness for that service is called objective trust. For some other services, their quality cannot be objectively measured. Sometimes, it largely depends on each individual’s taste and other subjective factors. In this situation, it is only meaningful to discuss the trustworthiness of the entity from the specific source’s point of view. This type of trust is classified as subjective trust [108].

Transaction based vs. Opinion based

If trust function relies on the information of individual transactions to infer an entity's trustworthiness, then trust function is called transaction based. But if it requests opinion information from others to infer trustworthiness, then it is called opinion based.

Complete vs. Localized information

Trust functions can also be classified according to the way information is collected. If trust function assumes that every entity has same access to all the transaction or opinion information, then trust function is classified as global trust function but if each entity has access to different information, based on its location or position in the environment, then trust function is categorized as localized trust function. A localized trust function is thus also subjective [108].

Rank based vs. Threshold based

If the trustworthiness returned by a trust function can be interpreted as an approximation of some of the property, on which threshold value can be defined, then trust function is called threshold based. But sometimes trustworthiness of a single entity alone does not convey much information. It becomes meaningful only when it is compared with the trustworthiness of other entities. Such trust functions return the relative ranking of an entity. These types of trust functions are classified as rank-based functions.

Types of Trust

As defined in [106], trust is classified into two categories: Identity trust and Behavior trust. Identity trust is concerned with verifying the authenticity of an entity to determine its trust level with respect to what the entity is authorized to do and what can be expected from it. Behavior trust, on the other hand, is more real and deals with a wider notion of entity's trustworthiness.

2.4.2 Approaches used for managing trust

Trust management is the activity of collecting, encoding, analyzing and presenting evidence relating to competence, honesty, security or dependability with the purpose of making assessments and decisions regarding trust relationships [102]. A trust management system for large scale distributed systems like grid and web should be scalable, reliable and secure. Two main approaches, currently available for managing trust are: i) Policy based trust management ii) Reputation based trust management. Following paragraphs briefly discuss each of these.

Policy Based Trust Management

Policy based trust management deals with using policy languages and engines for specifying and reasoning trust. The goal is to determine whether or not an unknown user can be trusted based on a set of credentials and a set of policies. The different entities or components constituting the system exchange and manage credentials to establish trust relationships which are further refined based on certain predefined policies. Policy based trust is involved in making access control decisions. It is intended for systems where behavior is guided by complex rules and policies.

Reputation Based Trust Management

In reputation based trust, the focus is on trust computation models capable of estimating the degree of trust that can be invested in a certain party based on the history of its past behaviors [102]. It is intended for distributed systems where a system only has a limited view of the information in the whole environment. New trust relationships and trust values are inferred based on the available information. The available information is based on the recommendations and experience of other users.

Another approach to manage trust is by using Trusted Authority (TA) where the basic concept is same as that of a CA (Certificate Authority) but are specifically meant for dealing with trust. But in large scale distributed computing system like grid, its credibility and usage decreases and uncertainty increases as the community of its trustees grows [109].

A trust management system generally includes negotiation of trust when a member joins, storage of trust metrics and distribution of trust metrics. The trust life cycle is composed of mainly three different phases: trust creation phase, trust negotiation phase and trust management phase. The trust creation phase generally is done before any trusted group is formed and it includes mechanisms to develop trust functions and trust policies. Trust negotiation is activated when a new un-trusted system joins the current distributed system or group. Trust management phase is responsible for recalculating, updating and storing trust values based on transaction information.

Trust model must provide mechanisms to express, evaluate and enforce trust related access control policies. These policies involve determining the trustworthiness of the target service/service provide/domain to reach at an authorization decision. Like privacy policies, most of the mechanisms used to express, evaluate and enforce trust policies are problem specific or not based on standards. WS-Trust describes a framework for trust models that enables web services to securely interoperate but it does not tell how to express, evaluate and enforce trust based access control policies. As already mentioned, XACML can be extended to express any general access control policy, it can be used to express trust related access control policies also.

2.4.3 Trust management in existing middleware

Much of the work to enable trust relationships in grid is through the use of X.509 based digital certificates which verify the identity of the requester. In this approach trust is assumed to be associated with the identity of the requester but this approach cannot be used to address complex trust related security issues. To address a wider notion of trust, a deeper understanding of interactions among service providers and service requesters is needed which cannot be achieved by identity based trust alone. So behavior trust should be used to address a wider notion of trust.

A number of important projects address the challenges related to trust management. Major among them include KeyNote [110], PolicyMaker [111], [112], PeerTrust [113], XenoTrust [114], NICE framework [115], Secure Grid Outsourcing (SeGO) [116], [117], TrustBuilder [118], [119], [120], [107], [109], [121] and [122].

Trust models such as KeyNote [110] and PolicyMaker [111] are concerned with identity trust. The trust model proposed by Mary R. Thompson *et al.* [112] describes trust issues involving reliance on Certificate Authorities and X.509 Identity Certificates and is also identity based. These trust mechanisms do not consider behavior trust which is dynamic and changes with time and thus these approaches have no mechanism to dynamically establish and monitor complex trust relationships. PeerTrust, XenoTrust, NICE trust management system and Secure Grid Outsourcing are reputation based trust management systems.

PeerTrust [113] was developed at Georgia Tech with Peer-to-Peer based electronic applications in mind. The PeerTrust metrics are derived from a combination of parameters like feedback, total number of transactions, factor of credibility, transaction context and community context. PeerTrust does not use a centralized database for storing the trust information. Rather, the trust information is stored in a distributed manner over the network. Each peer or a node in the network has i) a trust manager that is responsible for feedback submission and trust evaluation, ii) a small database that stores a portion of the global trust data, and iii) a data locator for placement and location of trust data over the network. The trust computation is based on the computation of trust data collected from different nodes or peers.

XenoTrust [114] is another trust and reputation management architecture. It consists of three main components: XenoServer, XenoCorp, and XenoServer Information Services

(XIS). XenoServers provide services to the client. XenoCorp provides authentication, auditing, charging, and payment services. XIS is used for storing the XenoServer status updates. The XenoTrust architecture introduces two levels approach of managing trust, called authoritative and reputation based. Authoritative trust is a boolean property established between a XenoCorp and the clients and servers that register with it. Reputation-based trust, on the other hand, is a discrete continuous property which quantifies, in a particular setting, the trustworthiness that one component ascribes to another [48]. It is distributed and highly subjective, in the sense that each entity has its own, independent view of others' reputations.

NICE [115] is a platform for implementing cooperative applications over the Internet. NICE uses a distributed trust evaluation mechanism. Whenever there is a transaction between two different nodes or peers, the peer receiving the service signs a cookie showing that the peer generating that service has successfully completed the transaction. Therefore, each node maintains a trust relationship and trust value based on the transaction it has received. The node generating transaction can store the signed cookies to prove its trustworthiness sometime later.

The Secure Grid Outsourcing (SeGO) system is developed for securely scheduling a large number of autonomous and indivisible jobs to grid sites. SeGO introduces fuzzy logic based trust integration model [116], [117]. SeGO scheme does not explicitly deal with the storage of trust information.

TrustBuilder [118] presents a policy language and infrastructure for trust negotiation for open systems. In TrustBuilder protocol and architecture, the negotiating parties establish trust between themselves by negotiating trust in a need-to-know manner. In this way, all the credentials are not disclosed to the either party. As it is a policy based trust management system it does not make use of any trust function to calculate trustworthiness of different parties. [119] proposes a general-purpose, application-independent Dynamic Distributed Trust Model (DDTM). In this, access rights are directly associated with a trust value which are further classified into direct trust values, indirect trust values and trust authorization levels. [120] is a work to establish a decentralized trust model. In this, the protocols used to disseminate trust are proposed and the algorithms used to update trust are described. [119] and [120] do not describe how trust policies should be expressed, enforced and integrated with other types of access control policies.

Other models that support behavior trust based on experience and reputation are proposed in [107], [109], [121] and [122]. Alfarez Abdul-Rahman *et al.* [107], [109] discusses a trust model that is grounded in real-world social trust characteristics, and is based on a reputation mechanism. The proposed model allows agents to decide which other agents' opinions they trust more and allows agents to progressively tune their understanding of another agent's subjective recommendations. The goal of this work is to provide a trust model for virtual communities that assist users in identifying trustworthy entities and gives artificial autonomous agents the ability to reason about trust [107], [109]. Some of the other trust models specialize in applying trust for enhancement of resource allocation functions. Farag Azzedin *et al.*[121], [122] present a trust model for grid systems and show how the model can be used to incorporate the security implications into scheduling algorithms. The proposed TRMS (Trust Aware Resource Management System) allocates resources considering a trust relationship between the resource provider (RP) and the resource consumer (RC) [122].

All the models described above are really a key inspiration of our work but these models do not address how to express trust policies and the integration of trust model with authorization framework.

Other proposed security models for grid like [24] and [43] do not deal with trust explicitly and does not provide any trust model to express, validate, update and manage trust relationships. Trust relationships implied from these security models are static and limited. Services like CAS [123] and VOMS [124] address general policy based authorization issues but do not make use of any trust model to express trust policies and to base authorization decisions on it. Similarly the specifications like WS-Trust [35] and WS-Federation [38] provide methods for issuing, renewing and validating security tokens and ways to establish, access the presence of and broker trust relationships but do not give any comprehensive trust model to determine the trustworthiness of different entities in a grid environment.

In the security policy framework, we have made an attempt to define a general trust model that deals with behavior as well as identity trust and discussed its integration with the authorization framework. The model also provides the mechanisms to express, evaluate update and enforce trust relationships and access control policies. It also describes the integration of trust based access with authorization framework.

2.5 Authorization

Authorization, in simple terms, deals with issues like who can access what services/ resources under what conditions. An authorization system can be defined as a system that grants specific type of access to specific requesters based on their authentication, what services/resources they are accessing, current state of the system and their conformation to established authentication, privacy, trust and other security policies. It is a detailed description of all aspects of a system dealing with access of services/resources by requesters. Grids present several unique authorization related challenges. There are several factors that make authorization hard in grid systems *e.g.* user population and resource pool is large and dynamic, resources have different authentication, privacy, trust and authorization requirements, computations span over multiple domains, users have different roles/privileges in different domains *etc.* All these factors make authorization a big and challenging issue. Following paragraph presents the authorization requirements of a grid system.

2.5.1 Authorization Issues

A grid system consists of virtual organizations. Virtual organizations consist of one or more physical organizations or administrative domains. These organizations provide services/resources that can be accessed by subjects of other organizations based on their authorization status. A number of different types of policies can exist among subjects and services in such an environment [125]. If a subject is a member of several domains then his/her access rights can be different in different domains. In order to access a service/resource, the requester must conform to the set of rules/requirements/policies defined by that service/resource. Every administrative domain has its own authentication, privacy, trust and authorization policy and the domain can change it dynamically. So access control in grids need to support multiple security policies and should have the flexibility to support dynamic changes in security policies [126]. In grids, the access depends on many factors like authentication requirements of the service, trust relationship with the requester, privacy requirements of the requester, authorization and other security policies among service providers and requesters *etc.* Authorization in such an environment should be determined as a result of evaluating the request of an authenticated user against various policies like privacy policy, trust policy, authorization policy *etc*. Many authorization mechanisms for large scale distributed systems like grid and web ignore one of the components from privacy, trust and policy. We emphasize that these are vital components and must be integrated into the access control framework of security architecture to make it more effective and useful [127].

Another desirable feature of authorization mechanism is to support fine grained access to shared resources/services [61], [128]. Support must be there for a flexible delegation mechanism also so that resources/services can be accessed on behalf of a particular user. Scalability and interoperability are also the important requirements that must be addressed by authorization system [48]. The authorization framework needs to be fully distributed also and it should be able to express VO wide, local site wide and other access control security policies related to grid as a whole.

2.5.2 Approaches used for Authorization

There are two general approaches for authorization: identity based and token based [47]. In identity based approach, only the authenticated identity along with the requested action is presented to the resource which then checks an internal list of allowed identities and actions list. If the authenticated identity and requested action appears in the internal list of allowed identities and actions then access is granted otherwise denied. Token based approach is also referred to as capability based authorization. In token based approach, a token (or authorization credential) is granted to the user who then presents it to the service as a proof of his/her rights. The service does not care who the presenter is, rather just that the request came with the appropriate token. The drawback of token based approach is that it is difficult to dynamically revoke access rights whereas in identity based approach, it is difficult to support delegation. Authorization systems can also be categorized as i) pull, push or agent based authorization systems and ii) VO level or resource level authorization systems. Following paragraphs briefly discuss each of these.

Push, Pull and Agent based Authorization

An authorization system where the authorization credentials are pushed to the access controller are called push based authorization systems. The credentials are first obtained by users from the relevant authority and then these are presented to the system to obtain access to resources. All certificate based systems employ this mechanism. In pull model, the users provide the minimum credentials to the access controller and it is the responsibility of the controller to check the validity of the user based on the policies of the system. The file systems in different operating systems employ pull mechanisms, where users provide their minimum credentials and access policies corresponding to the user are pulled by the operating system. Based on the policies, the operating system either allows or disallows the user to access the resources [48]. Push based mechanisms are more scalable whereas pull based mechanism are more user friendly as users do not have to obtain credentials from the relevant authority. Figure 2.4, Figure 2.5 and Figure 2.6 shows the difference between push, pull and agent based authorization systems.

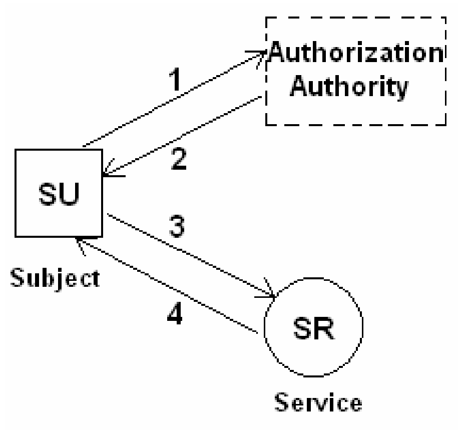


Figure 2.4: Authorization Push Sequence

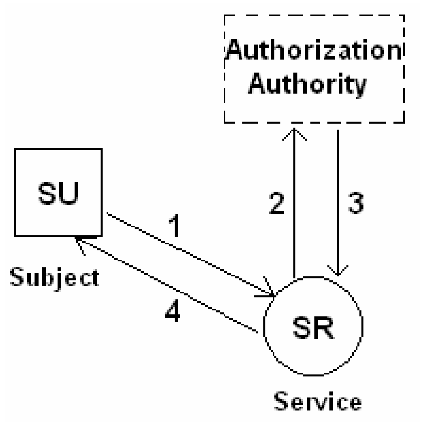


Figure 2.5: Authorization Pull Sequence

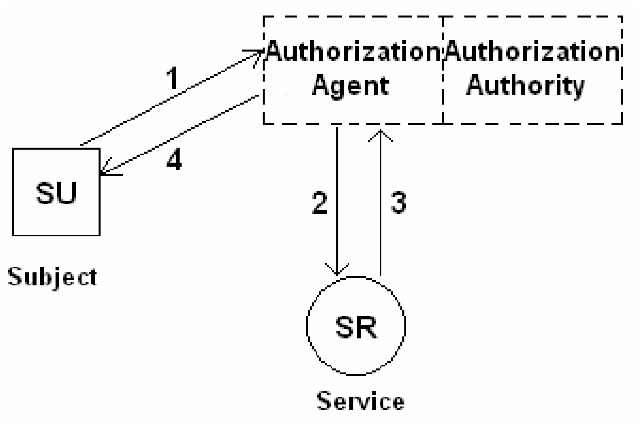


Figure 2.6: Authorization Agent Sequence

In agent based authorization, Subject contacts a higher level agent with a request to obtain service authorization. The agent then makes an authorization decision based on the rules established by authorization authority and if successful, it allows Subject to directly interact with the service [129].

VO Level and Resource Level Systems

VO level grid authorization systems are centralized authorization for an entire virtual organization. These types of systems are necessitated by the presence of a VO which has a set of users, and several resource providers who own the resources to be used by the users of the VO. Whenever a user wants to access certain resources owned by a resource provider, he/she obtains a credential from the authorization system which allows certain rights to the users. The user presents the credentials to the resource provider to gain access to the resource. The Resource Level authorization systems implement the decision to authorize the access to a set of resources. These mechanisms work at the resource level rather than at VO level. VO level and resource level authorization systems look at two different aspects of the grid authorization. These two authorization systems complement each other, and can be implemented together to provide a holistic authorization solution [48].

2.5.3 Authorization in existing middleware

A lot of projects like Akenti [130], CAS [123], PERMIS [90], CARDEA [131], PRIMA [89], GridShib [132], [133], GUMS [134], SESAME [135] and VOMS [124] are developed to address authorization related issues in one or other form in distributed environments but they have their limitations also. CAS and VOMS are VO level authorization systems whereas Akenti and PERMIS are resource level authorization systems. GUMS is grid user management system. Following paragraphs briefly discuss each of these.

Akenti [130] is an access control mechanism that uses digitally-signed certificates to define and enforce an access policy for a set of distributed resources that have multiple, independent and geographically dispersed stakeholders. The stakeholders assert their access requirements in use-condition certificates. Users are identified by X.509 identity certificates. During a request to use a resource, a policy engine collects all the relevant certificates and decides if the user satisfies all the requirements [130]. Akenti allow different stakeholders to express policies specifying how resources can be accessed but the policy language is different from XACML. Akenti policy is expressed in XML and stored in three types of signed certificates: policy certificates, use-condition certificates and attribute certificates [130]. Thus Akenti policy language model is different from XACML policy language model.

CAS [123] is an approach to the representation, maintenance, and enforcement of policies and provides a scalable mechanism for specifying and enforcing these policies. This approach 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. The owners of resources grant access to a community account as a whole. The CAS server is responsible for managing the policies that govern access to a community’s resources. It maintains fine-grained access control information and grants restricted GSI proxy certificates to the users of community [136]. CAS can also be used for managing role-based sub groups of a virtual organization [137]. CAS also support mechanism to delegate permissions on a set of resources distributed across different administrative domains but it focuses on centralized specification of community policies governing access to collection of resources. Moreover, in CAS, policies are not expressed using XACML.

PERMIS [90] is a policy driven RBAC Privilege Management Infrastructure (PMI). PERMIS implements role based access control (RBAC) scheme in which rights are associated with roles rather than with specific entities. PERMIS describes a policy driven role based access control system. The user's roles and the policy are stored in X.509 Attribute Certificates. The policy, written in XML, describes who is trusted to allocate roles to users, and what permissions each role has. Access control decisions are made by an Access Control Decision Function. The decision is made according to the requested mode of access, the user's trusted roles and the policy [90]. In PERMIS, policies are written in XML. We are using XACML to express policies which is OASIS standard.

Cardea [131] is a distributed authorization system that facilitates dynamic access control. It is developed as part of the NASA Information Power Grid [138], which dynamically evaluates authorization requests according to a set of relevant characteristics of the resource and requester rather than considering specific local identities. Potentially accessed resources within an administrative domain are protected by local access control policies, specified with the XACML [41] syntax, in terms of requester and resource characteristics. 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. Cardea is currently implemented in the Java language. In Cardea, policies are defined with respect to high level identities such as entity's distinguished name. The authorization decisions depend heavily on the attributes a service requester holds.

PRIMA [89] focuses on the issues of management and enforcement of fine-grained privileges. PRIMA mechanisms enable the use of fine grained access rights and adhoc and dynamic collaboration scenarios. PRIMA provides tools for end users and administrators to manage privileges for the resources they are authoritative for through X.509 Attribute Certificates that carry privilege and policy statements. An access control decision function in the form of an authorization module for the Globus Toolkit® authorizes requests based on the combination of user access privileges with resource policies and provisions low-level access control enforcement functions with decision qualifications [89]. PRIMA system is particularly motivated by the desire to support spontaneous, short lived collaborations among small group of grid users.

GridShib [132], [133] is NSF funded project between NCSA and the University of Chicago. The goal of the project is to integrate GSI [76] and Shibboleth [88] to provide the needed capabilities for a robust attribute infrastructure. This project will deliver a framework that allows participants in multi-organizational collaborations to control the attribute information that they want to publish, share, and reveal to other parties. Those parties will also be able to determine whether they possess the capabilities to access a service by matching their capabilities with the service's shared policy of required attributes. The pseudonymous interactions will be supported through the use of anonymous public key credentials that are mapped to the client's identity at the client's own discretion [74].

GUMS [134] is another system that automates user registration and management for a local grid site. The model allows users to register once with a VO and provide all the computing sites the information they need with the required level of trust. Tools have been developed to allow sites to automate the management of local accounts and the mapping between grid identities and local accounts [134]. GUMS is not an authorization solution instead it is a user management system.

SESAME [135] is dynamic context-aware access control mechanism for pervasive applications. SESAME complements current authorization mechanisms to dynamically grant and adapt permissions to users based on their current context [135]. The proposed mechanism extends the role based access control (RBAC) model while retaining its advantages. The model dynamically adjusts Role Assignments and Permission Assignments based on context information. Each user is assigned a role subset (by the authority service) from the entire role set. Similarly the resource has permission subsets for each role that will access the resource [135]. SESAME does not make use of any policy language like XACML to express access control policies.

VOMS [124] is a Virtual Organization Membership Service to manage authorization information in Virtual Organization scope. The VOMS System consists of four main components namely User Server, User Client, Administration Client and Administration Server. User Server receives requests from a client and returns information about the user. User Client contacts the server presenting a user's certificate and obtains a list of groups, roles and capabilities of the user. Administration Client is used by the VO administrators for adding users, creating new groups, changing roles *etc.* Administration Server accepts the requests from the clients and updates the database. The VOMS architecture uses the authentication and delegation mechanisms provided by the Globus Toolkit® Grid Security Infrastructure (GSI) [76], [124]. VOMS constitutes a system conceptually similar to CAS. It has a community centric attribute server that issues authorization attributes to members of the community.

There are many other attempts also like [139] - [156]. These approaches are mainly concerned with policy based authorization and access control but do not provide separate mechanisms for dealing with privacy and trust based access control. Xinwen Zhang *et al.* [139], [140] propose a usage control (UCON) based authorization framework for collaborative applications. Usage control policies are defined using subject and object attributes along with system attributes as conditions. The conditions can be used to support context based authorizations but the framework does not support obligations. Christian Schlager *et al.* [101] proposes authorization framework respecting privacy and flexibility in b2c eCommerce but it does not support context based authorization. The framework proposed by Jin Wu *et al.* [141] mainly supports fine grained access control for resource usage and management. Ludwiz Seitz *et al.* [142] present a system primarily enabling delegation using XACML access control system. Hai Jin *et al.* [136] proposes a universal, scalable authorization and access control framework called RB-GACA for grid systems but the framework does not provide facilities to integrate different authorization mechanisms. It is based on role based access control mechanism. Glenn Wasson *et al.* [143], [144] and Jun Feng [145] mainly focus on virtual organization wide resource usage and access control policy expression and enforcement mechanisms. Efforts like [146], [147], [148], [149] and [150] have their own mechanisms to express, evaluate and enforce access control policies and do not use XACML. The implemented authorization model is distinguishable from the discussed projects/models in different ways *e.g.* the implemented model supports fine grained and context based access. Policy expression is platform independent. The framework is able to express and enforce VO wide and service wide access control policies. It extends basic authorization mechanism to include trust and privacy based access to grid services. It also supports multiple security policies and provides facilities to integrate different authorization mechanisms.

Table 2.1 compares the existing middleware in terms of their support in the areas of authentication, privacy, trust and authorization.

Middleware / Projects	**Authentication**	**Privacy**	**Trust**	**Authorization**
CredEx	√	X	X	X
Entrust TruePass	√	X	X	X
Kerberized Certificate Authority	√	X	X	X
Liberty Alliance Project	√	X	X	X
Microsoft .NET Passport system	√	X	X	X
Microsoft Trust Bridge	√	X	X	X
MyProxy	√	X	X	X
Password Safe	√	X	X	X

EPAL	X	√	X	X
P3P	X	√	X	X
KeyNote	X	X	√	X
PolicyMaker	X	X	√	X
PeerTrust	X	X	√	X
XenoTrust	X	X	√	X
NICE Framework	X	X	√	X
SeGO	X	X	√	X
TrustBuilder	X	X	√	X
Akenti	X	X	X	√
CARDEA	X	X	X	√
CAS	X	X	X	√
PERMIS	X	X	X	√
PRIMA	X	X	X	√
SESAME	X	X	X	√
VOMS	X	X	X	√
Shibboleth	√	√	X	√
GridShib	√	√	X	√
CRISIS	√	X	X	√
Legion	√	X	X	√
GT4	√	X	X	√

Table 2.1: Comparison of existing middleware in terms of their support in the areas of authentication, privacy, trust and authorization.

In Table 2.1, middleware are marked under the column (authentication, privacy, trust and authorization) for which they provide the major support/system. This does not mean that these middleware cannot be used in conjunction with other middleware/mechanisms/technologies/systems to address other issues. *e.g.* Akenti has been marked under authorization column because it proposes and describes a specific authorization model and policy language. It has not been marked under authentication, though it support authentication through digitally signed certificates, because it does not proposes any new authentication model. Similarly other middleware like PERMIS, PRIMA and SESAME can authenticate users but marked under authorization column because more precisely these are authorization systems.

2.6 Problem Formulation

As observed during exhaustive review carried out in the previous section and summarized in Table 2.1, it is clear that grid services are continuously evolving, defining complex security requirements and policies between service providers and consumers. The multiplicity and complexity of security requirements and policies demand a security policy framework. Access to resources/services in a grid is provided based on conformance to established authentication, privacy, trust and authorization policies. These policies differ in their type, nature and scope. Most of the security solutions available do not provide comprehensive models to address different types of security requirements and access control policies. Many of them are not based on open standards. Some of them are problem specific or use proprietary mechanisms. As per Table 2.1, no middleware for grid services exist that integrates privacy, trust and policy based access control in a single authorization framework. Most of the existing middleware address one of the issues from authentication, privacy, trust or authorization and do not attempt to integrate all these in one common security framework. So a general, standards based integrated security policy framework is essentially required that addresses key security requirements related to authentication, privacy, trust and policy based authorization, and provides support to express, evaluate and enforce different types of access control policies in a uniform, integrated and platform neutral way. The integrated model will enable us to treat and specify the scenarios/issues concerning combination of authentication, privacy, trust and authorization as a single unit. Currently, there are very few initiatives towards defining such a framework. This thesis makes an attempt to define and implement such a framework that addresses important security requirements related to authentication, privacy, trust, authorization, and provide support to express, evaluate and enforce security policies related to them.

2.7 Objectives of the thesis

The first objective is to study and explore grid and web services to acquire basic knowledge about their manner of operation, execution, applications and differences between them. The second objective is to identify and categorize different types of security requirements and policies and to describe how security policies should be expressed and exposed. In the implemented framework, we have categorized security requirements and policies under four heads namely authentication, privacy, trust and authorization. Third objective is to define an integrated security policy framework for grid services. By security policy framework we mean the security architecture that addresses key security requirements of authentication, privacy, trust and authorization, and provide support to express, evaluate and enforce policies related to these requirements. The framework provides a set of features and services that tackle a set of security requirements and scenarios related to authentication, privacy, trust and authorization. Fourth and the last objective is to demonstrate the use and applicability of the framework.

To achieve the first objective, a comprehensive study of grid and web services architectures, their manner of operation, execution, applications and differences among them has been done. A thorough study of standards and specifications used in grid and web services based systems has also been carried out. Work done in the area of authentication, privacy, trust and policy based access in grid systems has also been studied in detail.

To achieve the second objective, the key security requirements and policies of grid systems have been categorized under four major heads namely authentication, privacy, trust and policy based authorization. Then the approach used for expressing and exposing these policies in the environment is discussed.

To achieve the third objective, we have implemented models for authentication, privacy, trust and policy based authorization and integrated them. Authentication Model provides support for single sign-on and delegation features through the use of proxy certificates. It also presents a credential management service to store, retrieve and update multiple user credentials. The privacy model provides support for anonymous access, hidden service access and access to private information based on conformance to privacy policies. The trust model provides support for calculating direct as well as recommended trust to determine trustworthiness of target service/resource. The policy based authorization model provides policy based access to grid services. The detailed explanation of these models and their integration is presented in the fifth chapter.

To achieve the fourth objective, the framework has been evaluated by implementing different security related scenarios and through implementations that involve enforcement of different types of access control policies. The implementation has been done in .NET environment with the support of WSE 3.0 toolkit.