TRUST Management for GRID Systems

Benjamin Aziz*, Alvaro Arenas*, Fabio Martinelli*, Paolo Mori*, Marinella Petrocchi*, Michael Wilson*
* STFC Rutherford Appleton Laboratory, United Kingdom
* Istituto di Informatica e Telematica (IIT), CNR, Italy

ABSTRACT

Grid computing is a paradigm for distributed computation on shared resources. It uses a large-scale, highly decentralized infrastructure, in which a huge number of participants share heterogeneous resources for a given purpose. Each participant both provides their own resources and exploits others' resources, combining them to solve their own problems. Trust management is a major issue in the shared Grid environment because Grid participants are usually unknown each other and usually belong to separate administrative domains, with little or no common trust in the security of opposite infrastructures. The standard security support provided by the most common Grid middleware may be regarded as one mean through which such common trust may be established. However, such security solutions are insufficient to exhaustively address all the trust requirements of Grid environments. In this chapter, we survey proposals for enhancing trust management in Grid systems.

KEYWORDS

Grid, Virtual Organizations, Trust Management, Reputation Management, Role-based Access Control, Supply Chain Management, Business Relationship Management, Trust-based Access Control.
INTRODUCTION

Innovations in information technology and business models are creating new security issues which require designs beyond those of traditional security solutions. In particular, the problem of guaranteeing that only authorized users have access to sensitive resources and data has been traditionally solved by adopting access control techniques. In these techniques the decision process is based on the identity or the role of the user. Since, in distributed environments with no central authority, the resource owner and the user that accesses the resource are often unknown to each other, traditional access control techniques cannot be applied. Consider an example in which a research institute adopts the policy of granting the right to execute applications on the computational resources shared by the institute, to professors of accredited universities. Although one authority may assert that the requestor’s identity is Alice Black, if this identity is unknown to the research institute, this does not help in making a decision whether she is entitled to use the resources or not. The crucial information needed in such scenario is the set of rights and qualifications of the requestor asserted by recognized authorities (i.e., the university she attends) together with trust information about the authorities themselves (is that university accredited for the research institute?). Trust management (Blaze et al., 1996), was born to implement distributed access control in decentralized systems, where access control decisions are based on statements called credentials made by multiple principals.

Grid is a distributed computing environment where each participant shares a set of his resources with others (Foster et al, 2001). This environment may group participants into virtual organizations. A virtual organisation is a set of individuals and/or institutions (e.g. companies, universities, research centres, industries and so on) who share their resources. A Grid user exploits this environment by searching among the available resources for a set that can be exploited to solve his problem. These resources are heterogeneous in that they could be computational, storage, software repositories and so on. The Open Grid Forum community has developed a standard to share resources on the Grid called the Open Grid Service Architecture (OGSA) (Foster et al, 2006), which defines the concept of Grid services and it is based on the Web Service Resource Framework (WSRF) (Banks, 2006). The Globus Toolkit 4 (Foster, 2005), is the reference implementation of the OGSA standard, and in this paper we refer to this implementation as the Grid environment (although the model developed applies to any possible implementation). Security is a very important issue for the Grid, because the participants are probably unknown to each other, and they belong to distinct administrative domains that adopt different security mechanisms and apply distinct security policies. Moreover, some participants can join or leave the virtual organisation during its lifecycle.

This chapter shows how trust management techniques can be successfully applied to support and enhance Grid security mechanisms. We will describe models, architectures, and implementations of trust management systems for the Grid, especially tailored for virtual organizations deployment. We review the existing models and the proposed architectures, as well as some prominent implementations for existing Grid toolkits (e.g., Globus). Both researchers and practitioners may benefit from this survey, which provides the state of the art at a glance and hints for future research and development.

The structure of the chapter is as follows. The next section gives an overview of the paradigm of virtual organizations, commonly used to model resource sharing in Grid systems. The third section gives an overview of some popular Grid security models and architectures as a means for establishing trust. In the following section, we discuss a couple of trust models and architectures; one for enhancing a role-based and trust management language with weights and the other for
utility-based reputation management in Grids. Finally, we conclude the chapter with a discussion on possible future trends for Grid trust management.

VIRTUAL ORGANIZATIONS IN GRID COMPUTING

Overview

Grid computing is a term often used to describe the amalgamation of several existing technologies such as cluster computing, Peer-to-Peer (P2P) computing and Web services technologies. In order to understand the behavior of such a heterogeneous blend of technologies, Grid systems are often modeled based on the paradigm of Virtual Organizations (VOs). In fact, the definition that Foster, Kesselman and Tuecke (2001) gave to the Grid in their article, *The Anatomy of the Grid*, is that Grid computing is concerned with “coordinated resources sharing and problem solving in dynamic, multi-institutional virtual organizations” (p. 2).

VOs are given attention by researchers within a wide range of fields, from social anthropology and organizational theory to computer science. Nevertheless, there has not been an agreed-upon definition of the concept, so the topic has been characterized by the amount of contributions, many of them are related to functional aspects, such as the role of IT in VOs, legal issues, social-economical aspects, etc. One such definition used in other projects (Wesner et al., 2006) is that “a VO can be seen as a temporary or permanent coalition of geographically dispersed individuals, groups, organizational units or entire organizations that pool resources, capabilities and information in order to achieve common goals”.

The parties that form a VO are typically part of a larger enterprise network or what is known as a virtual breeding environment within which the selection of partners is made in a phenomenon known as network activation in VO modeling theory (Camarinha-Matos & Afsarmanesh, 2003). The entities in the universe of such networks share some broad characteristics, e.g. belonging to the same economy or market sector, and their participation in the network indicates disposition to work together in a future market opportunity.

From the above definitions, it is clear that Grid computing supports the concept of VOs. In the beginning of Grid technology, Grid VOs generally consisted of supercomputing facilities with the aim of enhancing the computing power in order to perform very complex calculations in scientific environments. In the course of time, this situation has evolved, being possible today to connect different equipment in real time, according to the necessities of the applications and the resources available. Besides, it is possible to reallocate and to replace resources, to accommodate changes in requirements or to adapt to new opportunities in the business environment. This renders the nature of Grid VOs quite dynamic. This “dynamic nature” implies that the entire set-up of a VO may change in response to the market place. In this sense, VOs of this type are temporary as to their ability to react quickly as regards the membership, the structure, the objectives, etc. Its vague/fluid boundaries and opportunism, as well as equity of partners and shared leadership mainly characterize a dynamic virtual organization.

Virtual Breeding Environments

The concept of a Virtual Breeding Environment (VBE) was introduced by Camarilha-Matos & Afsarmanesh (2003) to model and support the rapid formation of virtual organizations, where a VBE may be regarded as an association of organizations adhering to common operating principles and infrastructure and who have as their main objective the participation in potential
VOs for the achievement of certain goals. Hence, we adopt the view that all VOs are formed within the scope of a more general VBE, as shown in Figure 1.

[file 1.tif]

Figure 1. The concept of a VBE

Organizations preregister to a VBE via a VO Manager service and this registration includes the details of resources and services they are willing to share in future VOs formed from the VBE as well as the list of potential users that belong to these organizations. A user willing to create a new VO assumes the role of the VO Owner, who with assistance of the VO Manager and other VO services (such as resource brokering services) is responsible for populating the VO with resources and users.

The VBE can be seen as a market place where resource providers compete to participate in VOs and users already in VOs compete for using resources. Reputation information about the resources and the organizations providing those resources could help guide VO Owners during their resource selection process. Similarly, reputation information of users registered with the VBE can be maintained, which would help VO resources providers in applying tight security measures for accessing their resources. Later, we discuss the definition of one such reputation model and its implementation that could maintain VBE-wide reputation values for both resources and their users.

VO Topologies

A VO, once it is formed out of a VBE, has a certain topology that may impact the trust and security issues within the VO. Such VO topologies have been discussed in literature (Burn et al., 1999; Katzy et al., 2000). For instance, Burn et al. (1999) define six types of VOs, ranging from organizations providing services in the web (such as web shops or newspapers on the web), which do not control any user of the service, to dynamic networks of entities collaborating to meet market opportunities. Some of the types of such VOs do not comply with our VO definition within the domain of Grid computing. Therefore, we focus here on three topologies for our VOs that were introduced initially by Katzy et al. (2000) based on network topologies.

Supply-Chain VOs

A supply chain in general may be defined as a coordinated system of organizations, people, processes and resources that moves information or services from one end called the producer (or supplier) to another end called the consumer (or customer). In our case, a supply chain consists of several organizations that are collaborating to achieve the goal of supplying the consumer with the end product, as shown in Figure 2.

[file 2.tif]

Figure 2. The Supply-chain VO Topology

VOs, which adopt a supply chain topology, often use supply chain management and efficient consumer response to improve inter-organizational co-ordination and control. Integration of information flow and material flow creates transparency in the entire value chain and reduces waste and doubles effort in the virtual enterprise.
Apart from the usual security issues related to supply chains, such as the authentication of participants, maintaining the integrity of the moving product and its desirable properties, ensuring the security of the product while in transit and auditing at each stage, Kang et al. (2001) identify some security issues related to inter-organizational workflows, which may be regarded as the generic form of the VO supply chain topology. These include separation of workflow-level security requirements from organization-level security, fine-grained and context-based access control and supporting dynamic constraints in order maintain desirable properties such as the separation of duties. The Usage CONtrol (UCON) model proposed by Park and Sandhu (2004) provides a means for solving the last two issues, since access is granted based on continuous monitoring of the behavior of the entity requesting the access to some object and dynamic context-based constraints form a part of this model.

**Hub-and-Spoke VOs**
In the hub-and-spoke topology (also known as the star topology), partners interact with one central hub or strategic centre, as shown in Figure 3. This type of VOs corresponds to a coordinated network of interconnected members, where each member provides key functionalities, and distinguished member plays the role of a leading actor (star), coordinating the whole operation of the VO.

![file 3.tif]

*Figure 3. The Hub-and-Spoke VO Topology*

Despite the decline in popularity of hub-and-spoke topologies in networks, they are still considered to be a good solution in VOs for large scale enterprise application integration problems. For example, having a central management unit facilitates the control of VO membership, even though this centralized management may become a performance bottleneck and a point of failure. Traditional Grid security solutions, as those advocated in GSI (Nagaratnam et al., 2002), have been focused on this type of topology.

**Peer-to-Peer VOs**
Pee-to-Peer topologies are characterized by the lack of hierarchy where any peer may interact directly with any other peer, as shown in Figure 4. The management of such VOs is usually based on self-organization.

![file 4.tif]

*Figure 4. The Peer-to-Peer VO Topology*

Wallach [Wal02] highlights a few security concerns in peer-to-peer systems. These include Secure routing, which ensures that when messages sent by non-faulty peers arrive at their non-faulty destinations without any compromises to their secrecy and/or integrity, Secure storage, where a node maintains the integrity of the data it stores and the data cannot harm the node (e.g. because the data contains a virus or a worm), distributed auditing, which is useful in resource usage monitoring and control and that could be related to other issues outside of security, as in load-balancing, and finally, trust and reputation, which are important factors in the security of Peer-to-Peer systems, in particular in identifying peers and evaluating their past behavior. Such an issue arises due to the lack of hierarchy in the topology.

**Trust-based VO Lifecycle**
The establishment of a VO from a VBE involves issues of trust and reputation in the sense that entities in a VBE have to communicate match-make and negotiate with other, possibly unknown, entities in the same VBE. Similarly, trust and reputation are needed during the operation of a VO during which entities belonging to trust domains that have particular reputation values related to their behavior history in the VO and the VBE attempt to access resources belonging to entities from other different trust domains. Once a VO is dissolved, the reputation that its entities have accumulated during its lifetime will be used to reflect their general VBE reputation. In general, a VO lifecycle demonstrates clearly the need for some notion of trust, either based on credentials or metrics or both. Finally, new organizations, resources, services and users joining a VO will have to demonstrate that they are trustworthy entities and that they will obey the VO policy during their membership.

The definition of a VO lifecycle that we adopt here is similar to that proposed by Strader et al. (1998) and adopted subsequently in several projects such as ECOLEAD (http://www.ve-forum.org/default.asp?P=284), TrustCoM (http://www.eu-trustcom.com/) and VOMAP (http://www.uninova.pt/~vomap/). According to this model, a VO lifecycle consists of the Identification, Formation, Operation and Evolution, and Dissolution phases, as shown in Figure 5.

Figure 5. The VO Lifecycle Phases

We discuss these phases more in detail next.

VO Identification
The identification phase addresses setting up the VO. This includes the selection of potential business partners from the network of enterprises by using search engines or looking up registries. Generally, relevant identification information contains service descriptions, security grades, trust and reputation ratings, etc. Depending on the resource types, the search process may consist of a simple matching (e.g., in the case of computational resources, processor type, available memory and respective data may be considered search parameters with clear cut matches) or of a more complex process, which involves adaptive, context-sensitive parameters. For example, the availability of a simulation program may be restricted to specific user groups or only for certain data types, like less confidential data, etc. The process may also involve metadata such as security policies or Service Level Agreement (SLA) templates with ranges of possible values and/or dependencies between them, such as bandwidth depending on the applied encryption algorithm. The identification phase ends with a list of candidates that potentially could perform the roles needed for the current VO.

VO Formation
After this initial step from the potentially large list of candidates, the most suitable ones are selected and to become VO members, depending on additional aspects that may further reduce the set of candidates. Such additional aspects cover negotiation of actual Quality of Service (QoS) parameters, availability of the service, “willingness” of the candidate to participate, etc. It should be noted that though an exhaustive list of candidates may have been gathered during the identification phase, this does not necessarily mean that a VO can be realized. For example, consider the case where a service provider may not be able to keep the promised SLA at a specific date due to other obligations.
In principle, the intended formation may fail due to at least two reasons: (a) no provider (or not enough providers) is able to fulfill all given requirements regarding to SLA, security, etc. or (b) providers are not (fully) available at the specified time. In order to circumvent these problems, either the requirements may be reduced ("choose the best available") or the actual formation may be delayed to be re-launched at a more suitable time. Obviously there may be the case where a general restructuring of the requirements leads to a repetition of the identification phase.

At the end of the (successful) formation phase the initial set of candidates will have been reduced to a set of VO members. In order to allow these members to perform accordingly their anticipated role in the VO, they need to be configured appropriately. During the formation phase, a central component called the VO Manager distributes the VO level configuration information, such as policies, SLAs, etc. to all identified members. These VO level policies need to be mapped to local policies. This might include changes in the security settings (e.g. open access through a firewall for certain IP addresses, create users on machines on the fly, etc.) to allow secure communication, or simply translation of XML documents expressing SLAs or Obligations to a product specific format used internally.

**VO Operation**

The operational phase could be considered the main lifecycle phase of a VO. During this phase the identified services and resources contribute to the actual execution of the VO task(s) by executing pre-defined business processes (e.g. a workflow of simulation processes and pre- and post processing steps). A lot of additional issues related to management and supervision are involved in this phase in order to ensure smooth operation of the actual task(s). Such issues cover carrying out financial arrangements (accounting, metering), recording of and reacting to participants’ performance, updating and changing roles and therefore access rights of participants according to the current status of the executed workflow, etc. In certain environments persistent information of all operations performed may be required to allow for later examination, e.g. to identify fault-sources.

Throughout the operation of the VO, service performance will be monitored. This will be used as evidence when constructing the reputation of the service providers. Any violation, e.g. an unauthorized access detected by the access control systems, and security threats, e.g. an event detected by an intrusion detection system, need to be notified to other members in order to take appropriate actions. Unusual behaviors may lead to both a trust reassessment and a contract adaptation. VO members will also need to enforce security at their local site. For example, providing access to services and adapting to changes and violations.

The evolution of the VO structure may be considered as part of the operational phase. This evolution is necessary since participants may fail all together or behave inappropriately and it becomes necessary to dynamically replace the misbehaving participants. This involves identifying new, alternative business partner(s) and service(s), as well as re-negotiating terms and providing configuration information as during identification, respectively formation phases. One of the main challenges involved with evolution is to re-configure an existing VO structure so as to seamlessly integrate the new partner(s), possibly even unnoticed by other participants. Ideally, one would like the new service to take over the replaced partners’ task at the point of its leaving without interruption and without having to reset the state of operation. There may be other reasons for participants joining or leaving the VO, mostly related to the overall business process, which might require specific services only for a limited period of time - since it is not sensible to provide an unused, yet particularly configured service to the VO for its whole lifetime, the partner may request to enter or leave the VO when not needed.
VO Dissolution
During the dissolution phase, the VO structure is dissolved and final operations are performed to annul all contractual binding of the partners. This involves the billing process for used services and an assessment of the respective participants' (or more specifically their resources) performances, like amount of SLA violations and the like. The latter may be of particular interest for further interactions respectively for other potential customers. Additionally it is required to revoke all security tokens, access rights, etc. in order to avoid that a participant may (mis)use its particular privileges. Generally the inverse actions of the formation phase have to be performed during the dissolution phase. Partial termination operations may be performed during evolution steps of the operation phase.

GRID SECURITY
Overview
Security techniques have traditionally been the main method by which trust is established in a Grid environment. The classical argument being that the more secure the system the more trustworthy it is. In this sense, the different security mechanisms utilized and shared by a number of organizations help establishing a trust domain around those organizations. An important step in the design of the security support for a given system is the definition of a threat model. A threat model is generally used to describe a given threat and the harm it could do a system if it has this vulnerability. The starting point to integrate a proper security support in the architecture of a complex system, such as the Grid, is the evaluation of a given threat model with respect to its benefits and costs.

In the literature, some threats models have been proposed, which are specific for the Grid system. Also, there exist numerous models, architectures, and implementations for enhancing the security of the Grid. In this section, after recalling three threats model, we provide an overview of the three most popular security architectures, namely the security model of the Open Grid Services Architecture (OGSA), the Grid Security Infrastructure (GSI) implementation of OGSA and the security architecture adopted by the Enabling Grids for E-sciencE (EGEE) project.

Threat Models for Grid
A threat model describes the potential for violation of a system, due to vulnerability that can be exploited to perform an attack to the system itself. Some specific threats models for the Grid environment have been proposed in the literature, see, e.g., (Naqvi & Riguidel, 2005), (Demchenko et al., 2005), and (Jiancheng et al, 2007).

As highlighted in (Naqvi & Riguidel, 2005), the main threats to a Grid environment are connected to its security requirements. Hence, they propose a model that identifies threats to: integrity of the physical infrastructure; confidentiality of the stored data; availability of the Grid resources; access control (i.e., authentication and authorization mechanisms).

Instead, (Demchenko et al., 2005) defines a different taxonomy that groups threats (and hence attacks) depending on their origin and the vulnerability they exploit. The authors identify the following attacks to a Grid system:

- The user credential attack, in which the user’s credentials are compromised, e.g., a credential theft.
- The wire intelligence attack, that can happen if service level communication is not protected enough against eavesdropping and interception;
• The malefactor initiated attacks. They are attacks executed by malicious users trying to harm the Grid services by traditional or web service specific attacks, e.g., by submitting forged XML requests, or by implementing denial of service attacks;

• The site management attack, caused by erroneous or improper configuration of the Grid security services and their management;

• The end service attack, which exploits known vulnerabilities of the specific services provided.

Starting from this classification, they propose a multilayer security framework that organizes a Grid node in security layers, or zones. For each of these zones properly security protections should be defined. However, a detailed description of the full approach is not given.

Finally, (Jiancheng et al, 2007) proposes a third taxonomy, which is similar to the previous one, based on the actors that we can identify in a Grid system. In particular, they define threats to

• Users, in terms of credential theft and compromises.

• Mediators, involving attacks to the communications between the user and the service provider

• Service providers, where malicious users exploit forged service requests to hang up the service engine.

• Resource providers, including attacks to the physical and software shared resources (e.g., exploiting configuration vulnerabilities, or performing an illegitimate use of the resource.

There are similarities between the last two models, because the four classes defined by the third model can be mapped on the first four classes of the second one.

**OGSA Security**

To address the Grid specific security requirements of OGSA, the OGSA Security Group proposed an architecture leveraging as much as possible from the Web Services Security specifications (Nagaratnam et al., 2002). The components of the architecture are shown in Figure 6.

![Figure 6. The OGSA Security Architecture (Adapted from Nagaratnam et al. (2002), p. 13)](file 6.tif)

As we mentioned previously, secure operation in a Grid environment requires that applications and services be able to support a variety of security functionalities, such as authentication, authorization, credential conversion, auditing and delegation. These functionalities are based on mechanisms that may evolve over time as new devices are developed or policies change. As suggested in (Siebenlist et al., 2003), Grid applications must avoid embedding security mechanisms statically.

Exposing security functionalities as services (i.e., with a Web Services Description Language (WSDL) (Christensen et al. 2001) definition) achieves a level of abstraction that helps provide an integrated, secure Grid environment. An OGSA infrastructure may use a set of primitive security functions in the form of services themselves. Nagaratnam et al. (2003) suggest the following security services:

**Authentication Service**
An authentication service is concerned with verifying proof of an asserted identity. One example is the evaluation of a User ID and password combination, in which a service requestor supplies the appropriate password for an asserted user ID. Another example involves a service requestor authenticating through a Kerberos mechanism, and a ticket being passed to the service provider's hosting environment, which determines the authenticity of the ticket before the service is instantiated.

**Identity Mapping Service**
The identity mapping service provides the capability of transforming an identity that exists in one identity domain into an identity within another identity domain. The identity mapping service is not concerned with the authentication of the service requestor; rather it is strictly a policy driven name mapping service.

**Authorization Service**
The authorization service is concerned with resolving a policy based access control decision. The authorization service consumes as input a credential that embodies the identity of an authenticated service requestor and for the resource that the service requestor requests, resolves based on policy, whether or not the service requestor is authorized to access the resource. It is expected that the hosting environment for OGSA compliant services will provide access control functions, and it is appropriate to further expose an abstract authorization service depending on the granularity of the access control policy that is being enforced.

**VO Policy Service**
The VO policy service is concerned with the management of policies. The aggregation of the policies contained within and managed by the policy service comprises a VO's policy set. The policy service may be thought of as another primitive service, which is used by the authorization, audit, identity mapping and other services as needed.

**Credential Conversion Service**
The credential conversion service provides credential conversion between one type of credential to another type or form of credential. This may include such tasks as reconciling group membership, privileges, attributes and assertions associated with entities (service requestors and service providers). For example, the credential conversion service may convert a Kerberos credential to a form that is required by the authorization service. The policy driven credential conversion service facilitates the interoperability of differing credential types, which may be consumed by services. It is expected that the credential conversion service would use the identity mapping service. WS-Trust ([http://docs.oasis-open.org/ws-sx/ws-trust/v1.3/ws-trust.html](http://docs.oasis-open.org/ws-sx/ws-trust/v1.3/ws-trust.html)) defines such a service.

**Audit Service**
The audit service similarly to the identity mapping and authorization services is policy driven. The audit service is responsible for producing records, which track security relevant events. The resulting audit records may be reduced and examined as to determine if the desired security policy is being enforced. Auditing and subsequently reduction tooling are used by the security administrators within a VO to determine the VO's adherence to the stated access control and authentication policies.

**Profile Service**
The profile service is concerned with managing service requestor's preferences and data which may not be directly consumed by the authorization service. This may be service requestor specific
personalization data, which for example can be used to tailor or customize the service requestor’s experience (if incorporated into an application which interfaces with end-users.) It is expected that primarily this data will be used by applications that interface with a person.

**Privacy Service**
The privacy service is primarily concerned with the policy driven classification of Personally Identifiable Information (PII). Service providers and service requestors may store personally identifiable information using the Privacy Service. Such a service can be used to articulate and enforce a VO's privacy policy.

**Grid Security Infrastructure**
The Grid Security Infrastructure (GSI) is a specific implementation of an OGSA-based Grid security architecture that has been packaged as part of the Globus Toolkit starting from version 2, GT2, up until the most recent version, i.e. GT4 (Globus Security Team, 2005; Welch et al. 2003). The GSI model in GT4 is shown in Figure 7 and is composed of the following security services.

![file 7.tif]

*Figure 7. The GSI Security Architecture (Adapted from Globus Security Team (2005), p. 2)*

**Authentication**
GSI defines a credential format based on X.509 identity certification, (Housley et al., 1999). An X.509 certificate, in conjunction with an associated private key, forms a unique credential set that a Grid entity (requestor or service provider) uses to authenticate itself to other Grid entities (e.g., through a challenge-response protocol such as TLS).

**Identity Federation**
GSI uses gateways to translate between X.509-based identity credentials and other mechanisms. For example, the Kerberos Certificate Authority (KCA) and SSLK5/PKNIT provide translation from Kerberos to GSI and vice versa, respectively. These mechanisms allow a site with an existing Kerberos infrastructure to convert credentials between Kerberos and GSI as needed.

**Dynamic Entities and Delegation**
GSI introduces X.509 proxy certificates, an extension to X.509 identity certificates that allows a user to assign dynamically a new X.509 identity to an entity and then delegate some subset of their rights to that identity.

**Message-Level Security**
Globus Toolkit Version 3 (GT3) and the subsequent version use the Web Services Security specifications (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss) to allow security messages and secured messages to be transported, understood and manipulated by standard Web services tools and software.

In relation to stateful and secured communication, GSI supports the establishment of a security context that authenticates two parties to each other and allows for the exchange of secured messages between the two parties. GT4 achieves security context establishment by implementing preliminary versions of WS-Security Conversation and WS-Trust specifications. Once the security context is established, GIS implements message protection using the Web Services standards for secured messages XML-Signature and XML-Encryption.
To allow for communication without the initial establishment of a security context, GT4 offers the ability to sign messages independent of any established security context, by using XML-Signature specification.

Trust Domains
The requirement for overlaid trust domains to establish VOs is satisfied by using both proxy certificates and security services such as CAS. GSI has an implicit policy that any two entities bearing proxy certificates issued by the same user will inherently trust each other. This policy allows users to create trust domains dynamically by issuing proxy certificates to any services that they want to interoperate.

EGEE Security Architecture
The EGEE (http://www.eu-egee.org/) security architecture (EGEE, 2004), shown in Figure 8 below, provides a number of security services including logging and auditing, authentication, authorization, delegation, data key management, sandboxing and site proxy, which can enhance the security of Grid access through EGEE’s Grid middleware called gLite (http://glite.web.cern.ch/glite/). The architecture is modular (modules can be plugged in and out of the architecture), agnostic (modules can evolve) and adherent to standards. In the following sections, we give an overview of the different security services of the architecture, although not all are yet implemented by EGEE.

Figure 8. The EGEE Security Architectures (Adapted from EGEE (2004), p. 11)

Logging and Auditing
The logging and auditing service is used to log information about time of security incidents, evidence of these incidents, produce incident reports and conduct security audits. Auditing on the other hand is not yet implemented, although the architecture defines it as a service that can be built based on the logging service.

Authentication
Authentication in EGEE’s security architecture is based on trusted third parties called Certification Authorities, which issue X-509v3 public-key certificates to express identity assertions. Authentication is achieved by verifying the distinguished name of the subject of the certificate. The service also facilitates single sign-on by allowing proxy certificates to be defined based on identity certificates. Revocation of certificates is done via periodic distribution of revocation lists.

Authorization
The model of authorization in EGEE’s security architecture is based on the push model, where the authorization service issues credentials to the Grid users, who present these credentials to the resources they wish to access. The architecture caters for two kinds of authorization services: attribute authorities and policy assertion services. Attribute authorities are implemented by the VO Membership Service (VOMS) (Alfieri et al., 2003) issuing X-509 certificates containing user role and group membership information. On the other hand, policy assertion services are implemented using the Community Authorization Service (CAS) (Pearlman et al., 2003) PERMIS (Chadwick & Otenko, 2002).
Delegation
The architecture provides for proxy certificates as a means for expressing delegation. The architecture still does not support the notion of least privilege, where an entity delegates the minimum set of rights required for achieving a task.

Data Key Management
The main aim behind the data key management service is to solve the problem of plaintext storage using the M-of-N storage technique. A full description of the problem is given in (EGEE, 2004).

Sandboxing
Sandboxing in the EGEE security architecture is achieved via tradition virtualization techniques, where services are run within well-isolated virtual machines, which provide users with complete operating systems.

TRUST AND REPUTATION IN GRIDS
Overview
The security models and architectures of the previous section help raise the trust level of any domain, however, as we mentioned in the section on VOs, VOs have a dynamic nature where new services and resources may need to be setup quickly and VO membership evolves frequently over time. In such VOs, security infrastructures adopted by existing members may need to be integrated and made interoperable with each others’ and new members’ infrastructures often in an automatic manner and within a limited amount of time. Hence, adopting trust models based on past behavior becomes necessary in fulfilling such demands of high dynamic nature, where the security-based models themselves are unable to fulfill.

In this section, we turn our attention to a couple of trust models and their architectures. The first proposes an extension to the Role-based Trust-management Mark-up Language (RTML) (Li et al., 2002; Winsborough et al., 2003) with weights denoting the trust that systems place on their users. These weights are used to make decisions about whether access can be granted or not for accessing system operations. The second model proposes a reputation management system for Grids based on utility functions, which express the satisfaction users get from their interactions with Grid resources.

Trust is a general concept with numerous definitions depending on the context in which it is used. In computer science, one of the most popular definitions was put forward by Grandison and Sloman (2000), who define trust as “… the firm belief in the competence of an entity to act dependably, securely and reliably within a specific context.” (p. 3). This definition encapsulates desired general properties of computing systems such as dependability, security and reliability while maintaining that these are only measurable within the specific context in which the system functions. For example, a system for establishing Web security will not be trusted in the context of the reliability of aerodynamics systems.

In Grid computing, as we mentioned earlier, trust has historically been focused on the strengthening of the security of Grid systems through adopting various flavors of authentication and authorization models. However, more recently, other architectures incorporating trust directly have been proposed and implemented based on measurable models (Dimitrakos, 2001; Lin et al., 2004, Hermoso et al., 2006; Vijayakumar and Wahida Banu, 2008).
Dimitrakos (2001) proposes a definition of trust within the context of service-oriented architectures, which also underline Grid computing. According to this definition, trust of a service requestor, A, to a service provider, B, in relation to some service, X, can be defined as follows: “Trust of a party A in a party B for a service X is the measurable belief of A in B behaving dependably for a specified period within a specified context in relation to X...” Interestingly, the model distinguishes between the notions of distrust and the lack of trust as follows: “Distrust of a party A to a party B for a service X is A’s measurable belief in that B behaves non-dependably for a specified period within a specified context in relation to service X.” Compared to the lack of trust, in this sense, distrust becomes a useful measure to revoke previously agreed trust, obstruct the propagation of trust, ignore recommendations and communicate a party’s trust value and whether this has reached the level of a blacklist for a class of potential business transactions.

Lin et al. (2004) develop a trust management architecture for trust enhanced Grid security, providing mechanisms for trust evaluation, recommendation, and update for trust decisions. The proposed formal trust model is capable of capturing the range of trust relationships that exist in a Grid computing system. Examples of such relationships are: Execution Trust that is the trust in the resource provider that will allocate the resources for the execution of the job request, and Code Trust that is the trust towards VO users that will not submit malicious job requests or malicious applications.

Hermoso et al. (2006) define a model of trust in VOs, which are a common abstraction of Grid computing as discussed in the previous section. The model is built on the notions of confidence and reputation of an agent within a role-based organization, which are measurable by values within the range [0,1]. The reputation of an agent is measured according to how well the agent performed in some role in the past. This will impact the confidence other agent’s have in this particular agent.

Finally, Vijayakumar and Wahida Banu (2008) propose a model of trust for resource selection in computational Grids. The model is based on assigning reputation weights to the different security capabilities that resources utilize and that reflect the degree of trust one can place upon these resources. For example, such capabilities include intrusion detection system capabilities, anti-virus capabilities, firewall capabilities, authentication mechanisms, secured file storage capabilities, interoperability and secured job execution. The values reflect how well these capabilities protect desirable Grid security properties such as confidentiality, availability and non-repudiation.

Other views of trust in Grid computing such as that of trustless Grid computing have also been proposed by Chang et al. (2002). The proposed model adopts the principle that trust (of a service being dependable, secure etc.) can only be attained through the use of rigorous mathematical proof. In this view, trust is not measurable; it is provable. The approach followed by Chang et al. (2002) is to use the proof-carrying code framework as first defined by Necula (1997), in order to assure the trustor of the validity of the claims made by the trustee. Proof-carrying code can be obtained as output from a special type of compilers known as certifying compilers. One big limitation of proof-carrying code-based approaches is that they are yet to be shown to be applicable in practice to large-scale systems such as Grids.

Another concept closely related to trust is reputation. Jøsang et al. (2007) define reputation as “what is generally said or believed about a person or a thing” (p. 622). Reputation is seen as one measurable means by which trust can be built, since one entity can trust (distrust) another based on good (bad) past experience and observations as well as collected referral information about its
past behavior (Abdul-Rahman & Hailes, 2000). In recent years, the concept of reputation has shown itself to be useful in many areas of research in computer science particularly in the context of distributed and collaborative systems, such as Peer-to-peer and Grid computing, Web services and social networks, where trust and security issues are strongly manifested. Interested readers may refer to the thorough survey of the state-of-the-art literature on reputation definitions, models and systems provided by Silaghi et al. (2007). Despite the fact that Globus offers a Monitoring and Directory Service (MDS) (Fitzgerald et al., 1997), the service lacks metric information, such as reputation, about resources that would reflect the quality of those resources. This poses a severe limitation on users of these resources especially during the formation and operational phases of the VO lifecycle. As a result, enhancing Globus with a reputation service would greatly benefit VOs in a similar manner that other distributed systems have benefitted.

Up to date, there has been only few reputation solutions proposed for Grid computing despite their clear advantage in enhancing the dependability of Grid systems through improving their resource allocation and scheduling operations. The GridEigenTrust model proposed by von Laszewski et al. (2005) exploits the beneficial properties of EigenTrust (Kamvar et al., 2003), extending the model to allow its usage in grids. A trust management system is integrated as part of the QoS management framework proposing to probabilistically pre-select the resources based on their likelihood to deliver the requested capability and capacity. The global trust for an organization with regard to another organization will be built from the direct trust that can be acquired from transactions between members of the two organizations and by considering trust information acquired from third party sources. The same trust aggregation scheme can be employed at the level of organization members, each of whom stores trust values for its transaction partners.

Another system is PathTrust (Kerschbaum et al., 2006), which is a reputation system proposed for member selection in the formation phase of a VO. To enter the VO formation process, a member must register with an enterprise network infrastructure by presenting some credentials. Besides user management, the enterprise network is provided with a centralized reputation service. At the dissolution of the VO, each member leaves feedback ratings to the reputation server for other members with whom they experienced transactions. The system requires each transaction to be rated by the participants.

**Role-based Trust Management with Weights**

The Role-based Trust Management framework RTML (Li et al., 2002; Winsborough et al., 2003) provides policy language, semantics, deduction engine, and concrete tools to manage access control and authorization in large-scale and decentralized system. RTML combines the strength of Role-Based Access Control (RBAC) and Trust-Management (TM). RBAC was developed to manage access control in a single organization in which the control of role membership and role permissions is relatively centralized in a few users. RTML takes from RBAC the notion of role to assign permissions to users. TM is an approach to distributed access control and authorization in which access control decision are taken on the base of policy statements made by multiple principals, e.g., Grid sites. From TM, RTML takes the principles of managing distributed authority through the use of credentials, as well as some notation denoting relationships between those authorities.

**The Model**

The main concept in RTML is the notion of roles: each principal has its own name space for defining roles, and each role is compounded by the principal name and a role term. For example, if $A$ is a principal and $r$ is a role term, then $A.r$ denotes the role $r$ defined by principal $A$. Only $A$ has the authority to issue policy statements defining the members of the role $A.r$. Roles may be
parameterized, e.g., a basic credential of the form \( A.r(p) \leftarrow D \) means that \( A \) assigns to \( D \) the role term \( r \) with parameter \( p \).

In the following credential, organization IIT assigns the role of IIT researcher to Alice, whose distinguished name adopted on the Grid is “CN=Alice, OU=IIT, O=CNR, L=Pisa, C=IT”.

\[
\text{IIT.researcher('CN=Alice, OU=IIT, O=CNR, L=Pisa, C=IT')} \leftarrow \text{Alice}
\]

In (Martinelli & Petrocchi, 2007) a basic set of RTML credentials has been enriched with trust, in order to express not only the fact that an authority assigns to someone a certain role, but also that a principal trusts someone for performing some functionality \( f \), or for giving a recommendation regarding a third party able to perform that functionality. This follows the interpretation of trust encoded in the transitive trust model of (Jøsang et al., 2003; Jøsang et al., 2006), according to which trust is always linked to a purpose. The most natural situation is when one trusts another for performing a certain function/task. It is often common that principals ask other principals for suggesting/recommending a third party able to performing that function or task.

The following language enriches RTML with trust, by also specifying a trust weight \( v \), i.e., a quantification of the confidence one places in the positive outcome.

**Simple member.** \( A.r(p,v) \leftarrow D \). The role \( A.r(p) \) has weight \( v \).

**Simple containment.** \( A.r(p,v) \leftarrow v_2 A_1.r_1(p_1,v_1). \) According to \( A \), all members of role \( A_1.r_1(p_1,v_1) \) with weight \( v_1 \) are members of role \( A.r(p,v) \) with weight \( v = v_1 \otimes v_2 \). \( v_2 \) is a constant filtering \( A_1 \)’s authority with \( A \)’s authority.

**Linking containment.** \( A.r(p) \leftarrow A_1.r_1(p_1).r_2(p_2). \) If \( B \) has role \( A_1.r_1(p_1) \) with weight \( v_1 \) and \( D \) has role \( B.r_2(p_2) \) with weight \( v_2 \), then \( D \) has role \( A.r(p) \) with weight \( v = v_1 \circledast v_2 \). This works as a sort of role-based delegation.

**Intersection.** \( A.r(p) \leftarrow A_1.r_1(p_1) \cap A_2.r_2(p_2). \) This statement defines that if \( D \) has both roles \( A_1.r_1(p_1) \) with weight \( v_1 \) and \( A_2.r_2(p_2) \) with weight \( v_2 \), then \( D \) has role \( A.r(p) \) with weight \( v = v_1 \circledast v_2 \).

Weights are not explicitly expressed in the linking and the intersection containment statements. Operators \( \circledast \) and \( \circledast \) combine the trust measures in the composed credentials expressed by the simple, the linking, and the intersection containment. For the sake of readability, weights are not explicitly expressed in the linking and the intersection containment statements. Generally speaking, \( \circledast \) combines opinions along a path, i.e., \( A \)’s opinion for \( B \) is combined with \( B \)’s opinion for \( C \) into one indirect opinion that \( A \) should have for \( C \), based on what \( B \) thinks about \( C \). \( \circledast \) combines opinions across paths, i.e., \( A \)’s indirect opinion for \( X \) through path \( \text{path1} \) is combined with \( A \)’s indirect opinion for \( X \) through path \( \text{path2} \) into one aggregate opinion that reconciles both. To work properly, these operators must form an algebraic structure called a c-semiring, (Theodorakopoulos & Baras, 2004).

**Extending the Globus architecture with RTML**

Security management in the Grid environment is complicated by the need to establish secure relationships between the VO members, among which no direct trust relationships may exist a priori, because during the VO formation phase, no trust relationships are required among the
parties that are joining the organization. The VO members share their computational resources for allowing the execution of unknown applications on behalf of other, possibly unknown, VO members. If an adequate security support is not adopted, the applications submitted by remote VO members during the VO operation phase could perform dangerous and even malicious actions on these resources. Hence, among the functionality of Grid security support, the authorization is a fundamental one.

In the following, we illustrate an architectural proposal for the enhancement of the security infrastructure of Globus (Foster, 2005) through the integration of the RTML framework. The proposed solution takes into account the level of trust that the VO member collected in her previous interactions with some Grid resources.

The standard authorization system provided by the Globus Toolkit is static, because the identity of each authorized user is statically mapped on a local account that is exploited to execute remote jobs on behalf of the user. Hence, given a Grid resource $R$, only users that have been previously registered on $R$ can access the services provided by $R$. This feature is a limitation in an open and distributed environment such as the Grid one.

Work in (Colombo et al., 2007a) proposes an authorization system based on RTML that determines the rights of a VO member on a Grid resource exclusively according to the attributes of the user. The system considers the complex trust relationships that the user collected in the past by exploiting services on other Grid resources/sites. These relationships are expressed in form of credentials, issued by the sites. From the Grid resource point of view, the credentials owned by a VO member determine attributes exploitable by the user on the resource. Each attribute corresponds to a set of privileges, and it is enforced through local accounts with the corresponding set of rights. The set of credentials is dynamic, because new credentials could be added by other Grid sites and some of the existing ones could be expired.

The proposed architecture is shown in Figure 9. When a VO member submits a request, the Globus container interacts with RTMLAuthzService (the authorization system based on RTML) through the Security Assertion Markup Language (SAML) (Madsen & Maler 2005) authorization Callout mechanism, to determine the rights of the user to perform the requested action on the requested service. The user request contains information about the user, the requested service and the repositories of the user’s credentials. Within RTMLAuthzService, the RTML Policy Decision Point is the component in charge of determining which rights the user holds on the requested service, according to her credentials and the access policy for that service.

In the current implementation, the RTML PDP downloads the user credentials from a public repository, and the access policy from a local repository. Both the user credentials and the access policy are translated into RT statements by the RTML parser. In its turn, the RTML Engine transforms each RT statement into Datalog statements, and returns them to the RTML PDP. A policy defines which attributes are required to execute the requested service and the PDP evaluates the Datalog statements through a Prolog engine to know which of these attributes could be granted to the user. The RTML PDP may compute more than one attribute for the user. In this case, if the user specifies the attribute he wants to exploit for the authorization process, the job will be executed with the privileges granted by that attribute. Otherwise, the RTML PDP chooses the attribute that grants the bigger set of rights to the user.

[file 9.tif]
Credentials and access policies as RTML statements

We present sets of user credentials that could be evaluated with respect to an access policy by the RTML authorization service in real job requests.

Trust Management example. The Center of Electronic Computation (CCE) offers a computational service to university students which are carrying out a stage at ABC Company. To access the service, a user needs to supply a credential issued by the ABC Company, asserting the stage, and a credential issued by the university, granting him the role of student. Also, a credential chain is required to verify that this university is admitted by CCE. Each student with right credentials can access the computational services, even if her identity has not been registered by CCE.

Let us suppose that Alice, which is a student of University of Pisa and a collaborator at the ABC Company, wants to access the CCE computational service. She supplies the following credentials:

1) UniPI.Student(university='University of Pisa', department='CS', id='1999s131', firstname='alice', lastname='black') ← Alice
2) ABC.Collaborator(role='stage', firstname='alice', lastname='black') ← Alice
3) MIUR.University(name='University of Pisa') ← UniPI

The Access Policy stored by CCE is the following:

a) CCE.University(name=?) ← MIUR
b) CCE.Student(university=ref{uni}, department=?, id=?, firstname=?, lastname=?) ← CCE.University(name=ref{uni})

CCE.ABCGuest ← ABC.Collaborator(role=?, firstname=ref{first}, lastname=ref{last}) ∩ CCE.Student(university=?, department=?, id=?, firstname=ref{first}, lastname=ref{last})
d) CCE.Guest ← CCE.Student(university=?, department=?, id=?, firstname=?, lastname=?)

Symbol '?' denotes a parameter whose value is not specified.

In the first and second credential, principals UniPI and ABC assign to Alice the attributes of, resp., student and collaborator, with the specified parameters. The owner of the third credential is UniPI and not Alice. This statement could be used to infer information about Alice's attributes. Indeed, rule a) of the access policy states that CCE considers as universities the principals that are considered Universities by MIUR, the Italian Ministry for University and Research.

The credentials and the access policy are retrieved by the RTML PDP from the repositories, and passed to the RTML Engine. The RTML Engine transforms them in a set of Datalog statements. Then, the RTML PDP reads from a configuration file that either attribute ABCGuest or Guest is required for the service requested by Alice, and invokes the Prolog Engine that evaluates the Datalog statements to verify whether Alice holds one of these roles.

Reputation Management example. Work in (Colombo et al., 2007b) exploits RTML to perform a fine grained control of the applications executed on Grid sites on behalf of remote VO members. With respect to the previous approach, this framework allows for determining not whether a certain user can execute an application on a Grid site, but it determines the allowed
actions that the application can perform during its execution. Moreover, this work extends the attribute-based trust management framework for the evaluation of the user’s credentials with a reputation management. Reputation of the VO member is calculated collecting past experiences encountered by Grid services with respect to that user. The more the user has well-behaved with a service, the more the service will positively recommend experiences with that user. The system architecture is shown in Figure 10. With respect to the previous approach, the RTML PDP is invoked at the level of the Globus Resource and Allocation and Management Service (GRAM), i.e., the service that allows the execution of applications on behalf of remote users. In particular, in (Colombo et al., 2007b), a behavioral PDP monitors the actions executed by the application, and invokes the RTML PDP when the security policy requires the evaluation of an attribute of the user, such as the reputation. For the sake of simplicity, here we simply show reputation predicates used to trigger the RTML PDP when the application tries to open the given file. An example of policy rule is

\[
\text{RepMaxOf(Unipi.files(user),0.8)}.\text{open(filename)}
\]

This means that the action \text{open(filename)} can be executed by the application only if the RTML PDP evaluates the predicate \text{RepMaxOf(Unipi.files(user),0.8)} to true, i.e., if the user is trusted to operate on files with at least weight 0.8.

\[\text{file 10.tif}\]

\textit{Figure 10. Interactions between Globus, GRAM, and the RTML Authorization Service (Adapted from Colombo et al. (2007b), p. 1506)}

Reputation credentials are expressed as RTML statements with weights. GRID services emit two kinds of credentials. The first kind expresses trust towards a functionality, e.g., towards good behaviors, \(A.f(v) \leftarrow D\), i.e., \(A\) trusts \(D\) for performing functionality \(f\) with degree \(v\). The others are credentials of recommendation, denoted as \(A.rf(v) \leftarrow D\). They express the fact that \(A\) trusts \(D\) as a recommender able to suggest someone for performing \(f\). Functionalities can be instantiated in several ways, e.g., \(A.files(p,v) \leftarrow D\). means that \(A\) trusts \(D\) with degree \(v\) for operating on a file.

As a simple example, imagine that Robert submits a request for executing an application on a Grid node that falls in the administrative domain of the University of Pisa UniPI. In particular, the application needs to operate on a file. UniPI considers as good recommenders the Institute for Informatics and Telematics and the University of Genoa. In order to evaluate the enforceability of the request, the RTML PDP will process the following credentials:

1) \text{UniGE.files('CN=B0b, 0.7')} \leftarrow Robert
2) \text{IIT.files('CN=B0b, 0.8')} \leftarrow Robert
3) \text{UniPI.rfiles('CN=UnivGenoa,1')} \leftarrow UniGE
4) \text{UniPI.rfiles('CN=InstInfTel,1')} \leftarrow IIT

The first two credentials say, respectively, that UniGe gives to Robert the attribute files (i.e., faculty of operating on files) with reputation 0.7 and that IIT gives to Robert the same attribute with reputation 0.8. The third credential says that UniPi accepts recommendations for the attribute files from UniGe. The fourth credential says that UniPi accepts recommendations for the attribute files also from IIT.
Credentials are evaluated with respect to a policy of this kind:

\[ \text{UniPI.files(username)}' \leftarrow \text{UniPI.rfiles(recname).files(username)} \]

The last statement expresses trust-based delegation. UniPI trusts a set of recommenders for choosing a third party trusted for the attribute \textit{files}. This means that, indirectly, UniPI trusts that third party for the attribute \textit{files}.

Combining the first and third credential with the policy, result is that UniPI trusts Robert for operating on a file with weight 0.7 \( \otimes 1 \) (0.7 when \( \otimes \) is the product operator), by means of UniGE’s recommendation. Similarly, combining the second and fourth credential, UniPI trusts Robert for operating on a file with weight 0.8, by means of IIT’s recommendation. The latter satisfies the previous reputation predicate.

The two previous approaches have been also merged to obtain a single framework that works both at coarse and fine grain level. Hence, RTML authorization service is used both to decide whether a VO member has the right to execute his applications on the Grid node and to control the actions that these applications perform while running.

We remark that the proposed approaches are useful to fight possible attacks defined in the threat model proposed by (Demchenko et al., 2005), namely, the Malefactor Initiated, the Site Management, and the End Service attacks. Indeed, the system does not allow untrusted users, i.e., the ones that most probably will perform attacks on the resource, to access the Grid Services.

**Utility-based Reputation**

Recently, utility functions have been used as the model basis for measuring reputation in Grid-based systems (Arenas et al., 2008). Utility functions are a means by which preferences can be expressed in a quantitative manner. In general, a utility function, \( f: C \rightarrow R \), maps each consumed entity, \( C \), to a real number \( R \) representing the satisfaction of the consumer from consuming that entity. Such utility functions can be used to capture the satisfaction of Grid users and resources based on their interactions with each other. For example, let’s first define the following functions, where \( \text{VOId} \) is the set of VO identities, \( \text{User} \) is the set of all users, \( \text{Res} \) is the set of all resources and \( \text{Org} \) is the set of organizations:

\[
\begin{align*}
\text{usersVO} : & \quad \text{VOId} \rightarrow \wp(\text{User}) \\
\text{resVO} : & \quad \text{VOId} \times \text{Org} \rightarrow \wp(\text{Res})
\end{align*}
\]

where \text{usersVO} expresses the set of users that a VO has and \text{resVO} the set of resources that an organization provides for a VO. Furthermore, let’s assume that

\[
\begin{align*}
\text{sla} : & \quad \text{VOUser} \times \text{Res} \times \text{VOId} \rightarrow R \\
\text{policy} : & \quad \text{VOUser} \times \text{Res} \times \text{VOId} \rightarrow \wp(\text{Action})
\end{align*}
\]

represents the expected quality of a resource (as formalized by an SLA) accorded between a VO user and the resource provider within a particular VO, and the normal behavior of the user (as formalized by a usage policy) again accorded with a resource provider within a VO and expressing the set of actions that user can perform on the resource.

A trusted monitoring service could then be used to capture interactions between resources and their users and based on these interactions, generate events as follows:

\[ \text{Event}_{\text{QoS}} = \text{VOUser} \times \text{Res} \times \text{VOId} \times R \]
\[ \text{Event}_{\text{Usage}} = \text{Res} \times \text{VOUser} \times \text{VOId} \times \text{Action} \]

where \( \text{Event}_{\text{QoS}} \) reports the real qualitative value, \( R \), generated by consuming a resource by its user within a VO and \( \text{Event}_{\text{Usage}} \) reports a prohibited action performed by a user on a resource within a VO.

Based on the above functions and events, one could then define a couple of utility functions expressing the satisfaction of users and resources as follows:

\[
\forall (u,r,\text{id},v) \in \text{Event}_{\text{QoS}} : \text{utility}_{\text{QoS}}(u,r,\text{id},v) = \begin{cases} 
1 & \text{if } v \geq \text{sla}(u,r,\text{id}) \\
\frac{v}{\text{SLA}(u,r,\text{id})} & \text{if } v < \text{sla}(u,r,\text{id})
\end{cases}
\]

\[
\forall (r,u,\text{vo},a) \in \text{Event}_{\text{Usage}} : \text{utility}_{\text{Usage}}(r,u,\text{vo},a) = \begin{cases} 
1 & \text{if } a \in \text{policy}(u,r,\text{vo}) \\
\text{penalty}(u,r,\text{vo},a) & \text{if } a \notin \text{policy}(u,r,\text{vo})
\end{cases}
\]

where \( \text{utility}_{\text{QoS}} \) expresses the QoS satisfaction of users from consuming a resource and \( \text{utility}_{\text{Usage}} \) expresses the satisfaction of a resource from its interaction with a user. The \( \text{penalty} : \text{VOUser} \times \text{Res} \times \text{VOId} \times \text{Action} \rightarrow [0,1) \) function computes the penalty each prohibited action by the user incurs, which is a value between 0 and less than 1.

The model of Arenas et al. (2008) then uses these utility functions to define a set of reputation functions for both the VO users and the VO resources, such that the reputation values are parameterized by the scope within which they occur. Hence for example, a resource is given the following reputation values: \( \text{Res}_{\text{rep eic}}, \text{Res}_{\text{rep ei}}, \text{Res}_{\text{rep e}} \) and \( \text{Res}_{\text{rep}} \). These express respectively the reputation of the resource arising from the satisfaction of a VO user in a particular VO, its reputation arising from the satisfaction of all users in the VO, its reputation in the VO and finally, its reputation in a VBE in general. Similarly, a user is assigned the reputation values \( \text{User}_{\text{rep eic}}, \text{User}_{\text{rep ei}}, \text{User}_{\text{rep e}} \) and \( \text{User}_{\text{rep}} \), which express respectively the reputation of the user arising from the satisfaction of a VO resource in a VO, its reputation arising from the satisfaction of all resources in the VO, its reputation in the VO and finally, its reputation in a VBE in general. Detail of the definitions of these functions is given in (Arenas et al., 2008).

A Reputation Management System for Grids

Based on the above model of utility-based reputation, a reputation management was proposed in (Arenas et al., 2008) and was implemented as a Globus service within the EU FP6 project GridTrust (www.gridtrust.eu). Figure 11 illustrates simplified system interfaces with other typical VO management services that may be running in any VO. These include a VO Manager (VOM) for managing issues such as VO membership and VO policies, a Reputation-aware Resource Broker (RRB) and a Resource Usage Control and Monitoring service (RUCM).

\[\text{Figure 11. A Reputation Management System for VOs (Adapted from Arenas et al. (2008), p. 5)}\]

The Reputation Management (RM) system accepts requests from the VOM for setting-up and terminating VOs through the following interfaces it offers to the VOM:

\[\text{setVO}(\text{VO ID}, \text{Resource ID List}, \text{User ID List})\]
where VO ID is the identity of the VO being set-up or terminated, Resource ID List is the list of resources of the VO, and User ID List is the list of users in the VO. The RRB service is used during the setting-up of new VOs by the VOM. During this phase, the RRB may request from the RM service the reputation of a resource in a particular VO or in the general VBE before proposing it to the VOM. This is achieved through the following RM interface:

Resource ID, Reputation Value: getResourceRep(Resource ID, VO ID)

where the returned values include the ID for the resource whose reputation is being requested and the value of that reputation. In case the VO ID is assigned a NULL value, the returned reputation will be the resource's reputation in the general VBE.

The RUCM service is a service that monitors requests and replies sent to and from a resource in its interaction with a VO user. The RUCM service can detect any undesirable behavior by the user in its usage of the resource being protected by that instance of the RUCM service. This is represented as prohibited actions such as the excessive storage of data on resources beyond the user's quota and the attempt to read or write unauthorized data. Such prohibited actions are reported through the following RM interface:

reportUser(Resource ID, User ID, VO ID, Action)

The invocation of this interface will cause with RM system to update the User_rep_eic, User_rep_ei, User_rep_e and User_rep discussed earlier in the section on the utility-based reputation model. The RM system can also accept ratings by VO users of QoS levels they have experienced in their interactions with VO resources. This is done through the following interface:

rateResource(User ID, Resource ID, VO ID, QoS Value)

which will cause the system to update the reputation values for resource, i.e. Res_rep_eic, Res_rep_ei, Res_rep_e and Res_rep. Finally, any of the entities in a VO may request from the RM service the reputation of a user:

User ID, Reputation Value : getUserRep(User ID, VO ID)

where the returned result includes the reputation and ID values for that user. Again, in the event that the VO ID is assigned a NULL value, the returned reputation will be the user's reputation in the general VBE.

Also in this case, the reputation management system addresses the threats that in the model defined by (Demchenko et al., 2005) have been classified as Malefactor Initiated, Site Management and End Service attacks. Indeed, this support prevents the Grid services to be used by VO users with a low reputation.

Usage Scenario
In this section, we demonstrate the applicability of the utility-based reputation management system of the previous section through an example of a usage scenario as depicted in Figure 12. In this scenario, we assume that the RRB starts by querying the RM system for the reputation of a couple of resources, Resource1 and Resource2, as part of the process conducted by the VOM of joining these resources to a new VO. If their reputation is satisfactory, the VOM signals to the
The RM system sets up the new VO and informs the latter of the two resources. It also informs the RM of the set of users who will be using the resources, which in this case we assume to be three, User1, User2, and User3. Once this setting-up of the VO is acknowledged by the RM system, the VO becomes operational and the users can avail of the resources offered.

During the operational phase of the VO, at some stage the RCUM service running locally at Resource1 captures a prohibited action performed by User3. It reports this action directly to the RM system. Based on the utility function for Resource1 and the penalty for the prohibited action, the RM system computes the satisfaction value of Resource1 in the context of the prohibited action and updates accordingly the different reputation values for User3. Sometime later the RCUM service for Resource1 requests to obtain the new reputation value for User3, for example, User_rep_eic. Based on this new value, RCUM may revise its decision to grant future accesses to User3 in order to use Resource1. This may or may not change the access rights for User3. Finally, the VOM decides to end the VO (as a result of achieving its goals) and informs the RM system of this decision. The RM system acknowledges this decision and deletes the VO entry from its database.

Figure 12. A Usage Scenario of the Reputation Management System (Adapted from Arenas et al. (2008), p. 6)

FUTURE RESEARCH DIRECTIONS
There are several areas of research that Grid trust management can be directed towards. The following sections outline some of these areas.

Comparison of Different Approaches
The different models and architectures we discussed in this chapter offer a variety of solutions to the problem of trust management in Grid systems. A comparison of the benefits and drawbacks of these solutions as well as the cost of adopting them would be one future area of research that could benefit the Grid community. There are several factors on which such a comparison could be based, which include usability entry level, cost of solution and its expressivity and maturity.

New Grid-related Paradigms
Due to the vast technological advances in computing systems and the necessity of adapting to dynamic business demands, new distributed computing paradigms have started emerging that require Grid computing to redefine and position itself in relation to these. Of the most popular of such current paradigms is Cloud computing (Boss et al., 2007; Jones, 2008). Cloud computing, stemming from virtualization techniques as well as utility and Grid computing, poses new trust and security management challenges due to the attractive simplicity of the model underlying it. A future research direction would be to investigate how the experience obtained from current trust management solutions could be applied to Cloud computing.

Threat Models for Trust Management
Another area for future research related to Grid trust management is constructing threat models. Threat models, such as threat trees, are used to uncover possible attacking routes that could be used by malicious users to attack the system. For example, one such famous threat in P2P and Grid systems is called collusion, where a number of resources collude to provide a fake reputation value for some other resource. Sabotage tolerance techniques (Sarmenta, 2002) can deal with such model of threats; however, other threats could be investigated in the future.
Quantitative Methods
Quantitative methods can provide a dynamic aspect to the definition of new trust management models and architectures, in which trust and reputation values would be traded-off against other quantitative aspects such as usage fees or costs of running applications on resources. For example, in such a cost-sensitive reputation model, the general welfare of users and resources in a VO can be measured not only based on the reputation of resources and users, but also with respect to the cost of using resources and the wealth of users. Highly reputable resources could be offered by their providers for usage in VOs for a high fee. Less wealthy users could then be eliminated from buying computational power on such highly reputable resources.

CONCLUSION
In the collaborative distributed computing environment of grids the management of trust is essential in order to allow the sharing of resources in a way that meets the security requirements of all those involved. In this chapter, we have presented an overview of some of the current models, metrics, architectures and implementations for trust management in Grid computing. The structure of a Virtual Organization (VO) has been used as an abstraction of collaborative activity on the Grid. We have described the trust challenges that need to be met at each phase of the VO lifecycle, and analyzed the security implications of adopting particular VO topologies. We then described existing Grid security architectures, including OGSA security, and implementations such as the Grid Security Infrastructure and the European EGEE grid security architecture. Trust management needs to be applied to whichever security architecture is adopted. Two alternative methods of trust management have been considered in detail: role-based trust management with weights, and utility-based reputation. There is no consensus on a solution to meet the security requirements of collaborative computing. Research efforts are still needed to relate trust mechanisms with security requirements on the one hand, and the organizational and legal constraints on collaboration on the other. The fashionable paradigm of cloud computing limits the risks that arise from sharing resources in grid computing, but it also has security requirements which will require trust management for collaborative work. Trust management is integral not only in these areas, but it will also be an essential part of further research into threat analyses for Grids, the analysis of coordinated attacks in Grids, and quantitative approaches to security in open systems such as Grids.

ACKNOWLEDGEMENTS
This work is partially funded under the IST program of the EU Commission by the project GRIDTrust, Trust and Security for Next Generation Grids, contract No. 033827, http://www.gridtrust.eu.

REFERENCES


