Computer Architecture Department

PhD. Thesis

"A Dynamic Validation Infrastructure for Interoperable Grid Services"

Jesús Luna García

Advisors:

Dr. Manel Medina Llinas

Dr. Oscar Manso Cortés

February, 2008

Abstract

Grid Resource owners can authorize access to their computing elements by means of well-established Authentication and Authorization processes for End-entities, through the use of cryptographic credentials that in most of the cases have a defined lifetime. Nevertheless, despite the fact that the adoption of Certification Authorities –CAs- has partially solved the problem of identification and authentication between the involved parties, and that Public Key Infrastructure –PKI- technologies are mature enough, we cannot make the same assumptions when untrusted domains are involved. Let us take for example two different Grid-CAs which do not have a direct interoperability agreement (i.e. explicit cross-certifying procedure), but their researchers need to work together. Furthermore to make this scenario even more complex, in the last years a lot of Grid Virtual Organizations -VOs- have been proliferating, each one usually installing its own Certificate Authority and thus giving birth to a large set of different and possibly untrusted security domains. This represents one of the biggest interoperability problems that could arise among all Grid users and therefore one of the major security challenges to be faced before building a widely distributed infrastructure allowing the cooperation of existing Grid installations.

There is a clear need of mechanisms for the computational Grid able to determine whether a particular end-entity's credential can be trusted at a given moment. This process is commonly named *validation* and traditionally it is performed via the placement of Certificate Revocation Lists (CRL) on a public directory accessible through protocols like HTTP. However this solution tends to be cumbersome for both the CA and the application. In terms of the CA it is difficult to manage because it involves providing revocation information efficiently (in some scenarios near real time notification is a must). Also on the client side such a solution penalizes efficiency, because it becomes forced to periodically download and parse the whole list of revoked certificates – which can be extremely large –in order to perform the validation process. In consequence, more efficient mechanisms to allow for the provision of real time certificate status information to relying parties have begun to be adopted in some demanding environments, where highly efficient and secure solutions are required. Among these available validation mechanisms, the Online Certificate Status Protocol (OCSP) stands out in the Grid community due to its ability to carry near real time certificate status information.

Despite its importance for security, OCSP not only faces considerable challenges in the computational Grid but also, in its current form, this protocol is unable to guarantee a *secure degree of interoperability* among all the involved Grid-Certification Authorities. Given the fact that interoperability among Virtual Organizations is becoming more and more important, the community is circumventing this problem using two main approaches:

1. Involved CAs *explicitly* build a trusted domain, defining a new CA hierarchy through cross certification techniques. In this case each CA explicitly trusts the others and therefore is able to accept their certificates.
2. Involved CAs do not build an explicit trusted domain, but interoperate through a "federation": any CA belonging to the federation implicitly trusts the others thanks to the definition of a well-established policy-based framework.

Even if the explicit trusted domain (first approach) is an attractive solution, it is not always possible to implement in Grid environments, because of the required agreements between the involved organizations, administrative overheads and technical problems that arise with current software.

For the computational Grid, the second of the aforementioned options (building a Federation of CAs) has been the most suitable solution for real-world projects so far. At this aim, the Policy Management Authorities (PMAs) represent "Federations of Grid PKIs" whose CA members accomplish minimum levels of security. These minimum requirements comprise the *PMA's Authentication Profile.* In the case of the existing Grid PMAs (America's TAGPMA, Europe's EUGridPMA, Asia-Pacific's APGridPMA and International's IGTF) compliance with their respective Authentication Profile is given through a well-defined -but mostly manual- process involving a careful analysis of the applicant PKI's Certification Policy -CP-, performed just once, when a new CA wishes to be part of an existing PMA. This is known as the PMA's *accreditation process*. Even though all the Grid CA members of a PMA must fulfill with the established authentication profile, not all of them accomplish these minimum requirements on the same level.
With independence of the interoperability mechanism chosen (explicit trust or CA-federation), any end-entity invoking a Grid Service's operation from the server, activates an authentication process that validates the end-entity's digital certificate according to the traditional *path validation* procedure.
When involved CAs interoperate thanks to explicit trust agreements, only *Basic Path Validation* is required: cryptographic verifications and status' checks over the involved certificates. State of the art Grid software like the Globus Toolkit, provides static mechanisms for the Basic Path Validation, i.e. the administrators manually declare the accepted CAs, and locally update respective CRLs. This is a cumbersome process in nowadays Virtual Organizations.
Therefore, despite the importance that an automated and complete security validation process has got in order to build dynamic trust relationships among Grid PKI's, to date there is no mechanism to automatically obtain this information.
This thesis presents a novel methodology and architecture for enabling *Extended Path Validation* in Grid environments for CAs that are part of the same PMA, thanks to the use of a *Validation Infrastructure* based on a Grid-enabled Online Certificate Status Protocol and, a *policy evaluation methodology* that compares the involved CAs' Certificate Policies to assert that they fulfil with a particular Authentication Profile and that they can therefore interoperate among them. The policy evaluation technique is based on a formal methodology originally proposed by researchers of the "Università di Napoli, Federico II" and the "Seconda Università di Napoli". A working prototype of the proposed Validation Infrastructure was also developed during our research, and is widely explained along this thesis.

Index

Index of Tables

Index of Figures

Acknowledgments.

Nunca hubiese llegado hasta este punto de no ser por el esfuerzo de dos personas que lo han dado todo por sacar adelante a sus hijos y han estado siempre conmigo, en las buenas y en las malas: mi padres, María Elena y Antonio. ¡A ustedes, gracias!

El esfuerzo de mis padres ha dado ya frutos muy valiosos en la figura de mis hermanas y mi hermano que me han apoyado en todo momento. Lilia, Lulú, Rosa y Toño ¡les doy también las gracias!

Una persona llena de cualidades, que me ha brindado su amor y llenado de su energía para seguir siempre adelante: Silke, ¡muchas gracias!

A ustedes, Miguel y Gabriel que creyeron en mi y me dieron su apoyo incondicional para que iniciase el Doctorado. ¡Gracias!

Quienes me han brindado no solamente su experiencia profesional y académica, sino también su amistad sincera estos años. Valentina, Manel, Oscar y Massimiliano. Moltes mercès! Grazzie!

Y a mis grandes amigos y familiares, con quienes he pasado momentos inolvidables: Nikolas, Elli, Alejandro, Angel, Jetzabel, Carlos Ll., Oscar C., Rubén, Javier R., David H., Lluis M. y muchos mas que siempre estarán en mi mente. A todos ustedes, ¡mil gracias!

Humanity needs practical men, who get the most out of their work, and, without forgetting the general good, safeguard their own interests. But humanity also needs dreamers, for whom the disinterested development of an enterprise is so captivating that it becomes impossible for them to devote their care to their own material profit.

Without doubt, these dreamers do not deserve wealth, because they do not desire it. Even so, a well-organized society should assure to such workers the efficient means of accomplishing their task, in a life freed from material care and freely consecrated to research.

Marie Curie (1867-1934) Physicist and chemist

Basic research is what I am doing when I don't know what I am doing.

Werner Von Braun (1912-1977) Rocket engineer

CHAPTER
1 INTRODUCTION

1.1 Grid and Web Services: a new paradigm

The past several years have seen a remarkable maturation and standardization of Grid technologies around the banner of the Open Grid Services Architecture (OGSA) as introduced in the "Physiology of the Grid" [OGSA], where was defined a clear separation between protocols and messages -mostly based on XML- required for interoperability among virtual organization (VO) components from the nature of the services responding to those messages. OGSA Grid technologies are based on a service-oriented architecture, where a service is defined as an entity that provides some capability to its clients by exchanging messages. By thus defining these *operations* only in terms of message exchange, it is possible to achieve great flexibility in how services are implemented and where they may be located. A service-oriented architecture is one in which all entities are services, and thus any operation visible to the architecture is the result of message exchange.

Therefore the Grid was envisioned as an extensible set of *Grid Services* able to be aggregated in various ways to meet the needs of VOs, which themselves can be defined in part by the services that they operate and share. OGSA is commonly referenced as the alignment of Grid and *Web Services*, defining a Grid Service as a transient, stateful and dynamically instantiated Web Service that provides a set of well-defined interfaces and that follows specific conventions.

In analogy to Web Services environments, a goal of the OGSA design is to allow common service behaviors (i.e. monitoring their status) to be expressed in standard ways regardless of context, so as to simplify application design and encourage code reuse. To achieve this reuse of behaviors, operations are often grouped together to form a service *interface*. Interfaces can then be combined –composed- to specify a service with the desired behaviors. By encapsulating service operations behind a common XML-based message-oriented service interface, service-oriented architectures encourage service virtualization, isolating users from details of service implantation and location.

Interaction with a given service is facilitated by using a standard *interface definition language* (IDL) to describe the service's interface(s). An IDL defines the operations supported by a service, by specifying the messages the service consumes and produces. An interface specification describes the messages the service expects but does not define what the service does in response to those messages -its *behavior*-. This behavior can be specified either informally through human-readable specification documents or via a formal semantics description. In either case, an important role of an IDL is to provide well-known names to which such semantics can be attached, in order to facilitate discovery of services that perform desired functions.

A well-defined interface definition language and a separation of concerns between service interface and implementation simplify the manipulation and management of services in four important respects:

- Service discovery: is important in distributed computing because it is frequent to operate in unfamiliar environments in which the identity and detailed characteristics of available services are unknown. In a service-oriented architecture, is easy to create registries containing information about the interfaces of available services, which users can query to find suitable candidates.
- Service composition: it enables code reuse and the dynamic construction of complex systems from simpler components. A well-defined IDL simplifies composition because a client need only know a service's interface to invoke it. Support for multiple protocol bindings can be important for service composition, by allowing

for optimizations designed to reduce the cost of service invocation within specialized settings.

- Specialization: refers to the use of different implementations of a service interface on different platforms. This strategy can facilitate seamless overlay, not only to native platform facilities, but also to virtual ensembles of resources via the nesting of service implementations.
- Interface extension: is an important feature of an interface definition language, as it allows for specialized implementations to add additional, implementation-specific functionality, while still supporting the common interface, This allows for the development of standard service interfaces, without requiring a "least common denominator" approach to implementation.

An important merit of this service-oriented model is that all components of the environment can be virtualized. By providing a core set of consistent interfaces from which all Grid services are implemented, it is possible to ease the construction of hierarchical, higher-order services that can be treated in a uniform way across layers of abstraction. Virtualization also enables mapping of multiple logical resource instances onto the same physical resource, composition of services regardless of implementation, and management of resources within a VO based on composition from lower-level resources. It is virtualization of Grid services that underpins the ability for mapping common service semantic behavior seamlessly onto native platform facilities.

1.2 Grid Services: eScience meets eBussiness

Nowadays there is no doubt about the importance of Grid Services, as they are bridging the gap between eScience and eBussiness by building cheaper and faster solutions. A clear example of this importance can be clearly seen into the workgroups members of the Open Grid Forum (OGF) [OGF], a new organization that resulted from the merger of the Global Grid Forum (GGF) and the Enterprise Grid Alliance (EGA).

The GGF grew out of a series of conversations, workshops, and Birds of a Feather (BoF) sessions that addressed issues related to grid computing. The first of these BoFs was held at SC98, the annual conference of the high-performance computing community. That meeting led to the creation of the Grid Forum, a group of grid developers and users in the U.S dedicated to defining and promoting grid standards and best practices. By the end of 2000, the Grid Forum had merged with the European Grid Forum (eGrid) and the Asia-Pacific Grid Forum to form the Global Grid Forum. The first Global Grid Forum meeting was held in March 2001. Since then, the GGF has produced numerous standards and specifications documents and held successful events around the world.

On the other hand, the EGA was formed in 2004 to focus exclusively on accelerating grid adoption in enterprise data centers. The EGA addressed obstacles that organizations face in using enterprise grids through open, interoperable solutions and best practices. The alliance published the EGA Reference Model and Use Cases, a Security Requirements document and Data and Storage Provisioning document. The EGA also significantly raised awareness worldwide of enterprise grid requirements through highly effective marketing programs and regional operations in Europe and Asia.

The efforts of the OGF to deliver Grid infrastructures to both, research centers and the industry, have raised multiple taskforces to fulfill in several cases sets of quite restrictive requirements (i.e. in eHealth environments, where privacy is a must).

A major concern related with Grid adoption is related with security, which plays a very important role in all the computational environments where a need for authentication, authorization, privacy, integrity and fault tolerance is a must.
For the research presented in this thesis, *Authorization* refers to the process of granting privileges to processes and, ultimately, users. This differs from *Authentication* in that authentication is the process used to identify a user. Once identified (reliably), the privileges, rights, property, and permissible actions of the user are determined by authorization. The next section will focus in these two of these aspects.

1.3 Authentication and Authorization for Grid Services

The problem of authentication (AuthN) and authorization (AuthZ) in computer systems and more specifically in distributed systems has been an area of growing interest these last years. Its beginnings can be distinguished since 1984, when Henry Levy introduced the concept of "capabilities" in computer systems [CapSys] making a through survey of early capability-based and object-based hardware and software systems. On his book, Levy defines a capability as a "token, ticket, or a key that gives the possessor permission to access an entity or object in a computer system" and as we shall see later, nowadays this term is still valid to define authorization schemes for a wide variety of distributed systems ranging from Web Services to the computational and data Grid. However we can find some interesting common properties shared between current security implementations, mainly related with the integration of the authentication and authorization processes in one system often referred as "Authentication and Authorization Infrastructure" or AAI as we shall call it along this document.
Many AAI implementations have developed secure mechanisms to express user's attributes and resource's use-conditions, in such a way that a decision capable engine may derive a correct, traceable, unambiguous and promptly response for an AuthN or AuthZ query. However the development of such a wide range of AAI solutions has caused a lot of interoperability issues and even "redundancies", where identical functionalities are being implemented with different technologies on the same distributed system thus resulting in performance problems and even in the impossibility to offer a consistent security level among them.
A new generation of security mechanisms is *policy-driven*, where *policy means a set of rules* that are used by the system to determine whether an authenticated subject can be permitted to gain access to a protected object. Computer systems must enforce a mandatory security policy that can effectively implement access rules for handling sensitive information. Base requirements for AAIs can be specified in frameworks describing basic architectural entities along with its trust relationships, message sequences and use case scenarios, privilege and policy management, and in some cases even enforcement mechanisms. In this section we introduce one of these frameworks specifically designed for Grid environments, which even though is more biased towards authorization, is important to review because of its relevance for the proposal presented in this document.
Despite the Grid authentication process is not directly referenced in the framework to introduce, it is providing most of the underlying security features related with the identity of involved participants. The following section will provide a quick review on the concepts of a widely used authentication mechanism for today's Grid installations: the Public Key Infrastructure.

1.3.1. Public Key Infrastructure in a glimpse

Public Key Infrastructure or PKI is the general term for a security infrastructure based on public-key cryptography. Such an infrastructure defines message formats, protocols and procedures that allow entitled to securely communicate claims and statements. The most commonly used assertions are those that bind identity, attributes, and authorization statements either to keys or to identities.

The most popular PKI is defined by the IETF's PKIX working group [RFC3280], which defines a security system used for identifying *relying parties* (users and resources) through the use of X.509 identity certificates [X509]. In this PKI, highly trusted entities known as Certificate Authorities (CAs) issue X.509 certificates where essentially a unique identity name and the public key of a client (also called *end-entity*) are bound through the digital signature of that CA. As a trusted third party, the CA can be used as an introducer: through the proof of private key possession and the validation of the CA's issued X.509 certificates, relying parties are able to associate a trusted, unique identity name with the communicated claims and statements of end-entities.

Although the use of PKI-related technologies has achieved a fair amount of adoption in the Web through the use of X.509 certificates for secured Web Servers, they have encountered some drawbacks in other scenarios. For example X.509 certificates' life cycle management (i.e. issue and revocation) has been a factor delaying its deployment in applications where the potential number of users is quite large (i.e. eGoverment). Besides, there are administrative and legal factors that need to be solved in order to avoid incompatibilities with national legislations; this is the particular case of privacy issues in eHealth systems based on PKI technologies.

The emerging XML-based security assertions formats and protocols are providing equivalent technologies to those defined by X.509/PKIX, with the added advantage of being more agnostic toward the underlying Public Key Infrastructure. The latter property will make it easier for organizations to choose or to migrate to public-key technologies that deploy alternative assertions formats and trust models. Having that flexibility is important to Grid security, as identities associated with services can be dynamically created and short-lived and may not have identity names associated with them –a situation that does not allow to fit properly the traditional X.509/PKI model with its centralized CA and focus on identity assertions-.

To provide an alternative that will address the dynamically created and short-lived identities within the X.509/PKIX framework, the Grid security community is actively involved in the specification of extensions to the X.509 certificate format to use these same certificates for the dynamic issuance of identities and delegation of rights by users [RFC3820]. The advantage of this approach is that those certificates can be used transparently in existing technologies that already adopted X.509/PKIX such as the transport layer security (TLS or SSL) [RFC2246].

In the following section we will present in more detail the underlying system called "Grid Security Infrastructure", an X.509-based mechanism that implements PKI authentication through the so-called Proxy Certificates in widely deployed Grid installations.

1.3.2. The Grid Security Infrastructure

The open source Globus Toolkit [Globus] is a fundamental enabling technology for the Grid, letting people share computing power, databases, and other tools securely online across corporate, institutional, and geographic boundaries without sacrificing local autonomy. The toolkit includes software services and libraries for resource monitoring, discovery, and management, plus security and file management. In addition to being a

central part of science and engineering projects that total nearly a half-billion dollars internationally, the Globus Toolkit is a substrate on which leading IT companies are building significant commercial Grid products.
Grid computing research has produced security technologies based not on direct interorganizational trust relationships but rather on the use of the Virtual Organization concept (VO) as a bridge among the entities participating in a particular community or function. The results of this research have been incorporated into the Globus Toolkit that uses public key technologies to address issues of single sign-on, delegation, and identity mapping, while supporting standardized APIs. The Grid Security Infrastructure (GSI) is the name given to the portion of the Globus Toolkit that implements the base security functionalities. GSI is a combination of various existing and custom developed security technologies that have been carefully integrated to support computational and data Grid projects.
During its first versions GSI was implemented on top of the Generic Security Services application programming interface (GSS-API) [RFC2078]. As the name implies, GSSAPI provided security services to callers in a generic fashion. These services could be implemented by a range of underlying mechanisms and technologies, allowing source-level portability of applications to different environments. GSS-API allowed constructing GSI simply by transcribing the grid security protocols into GSS calls. Thus it was possible to exploit various grid-level security mechanisms without altering the GSI implementation.
The recent definition of the Open Grid Services Infrastructure specification and other elements of the Open Grid Services Architecture [OGSA] within the Open Grid Forum, introduced new challenges and opportunities for Grid security. In particular, integration with Web services and hosting environment technologies introduced opportunities to leverage emerging security standards and technologies such as the Web services security.
Integration of GSI with OGSA enabled the use of Web services techniques to express and publish policies, allowing applications to determine automatically what security policies and mechanisms are required of them. Implementing security in the form of OGSA services allows those services to be used as needed by applications to meet these requirements. Advanced hosting environments enable security functionality to be implemented outside of the application, simplifying development.
In particular, GSI defines a cryptographic credential format based on X.509 identity certificates. An X.509 certificate, in conjunction with an associated private key, forms a unique credential set that a Grid end-entity (requestor and service provider) uses to authenticate itself to other Grid end-entities. As mentioned before, each GSI certificate is issued by a trusted party known as a Certificate Authority (CA), usually run by a large organization or commercial company. A CA acts as a root of trust for identity name assertions. In order to trust the X.509 certificate presented by an entity, one must trust the CA that issued the certificate. In most of the cases, this trust relationship is manually established and managed from inside the Globus Toolkit installation.
X.509 identity certificates are used within GSI because the establishment of trust between two administrative domains through this public key technology is relatively lightweight and unambiguous. Trust in a CA can be established unilaterally, which means that a single end-entity in an organization can decide to trust any CA without necessarily involving the organization as a whole. This is a key feature to the establishment of Virtual Organizations (VOs) that involve only some portions of the *real* organization, where the latter as a whole may provide little or no support for the VO and hence have little or no interest in trusting any CAs used by that VO.

GSI has been used, in turn, to build numerous middleware libraries and applications, which have been deployed in large-scale production and experimental Grids. GSI has emerged as the dominant security solution used by Grid efforts worldwide and this experience has proven the viability of using X.509 credentials as a basis for authorization, furthermore has allowed the use of a special subset of these, the so-called X.509 *Proxy Certificates*, as the mechanism for providing useful features like job delegation into the computational Grid. The following section explains in more detail these credentials.

1.3.3. Proxy Certificates

GSI introduces the X.509 Proxy Certificates (PC), an extension to X.509 identity certificates that allows a user to assign dynamically a new X.509 identity to an end-entity and then delegate some subset of their rights to that identity. Users create a proxy certificate by issuing a new X.509 certificate signed with their own private key instead of involving a CA. This mechanism allows new credentials and identities to be created quickly without requiring the use of a traditional administrator. Thus, this approach solves the issue of how to delegate rights to entities, both dynamic and static while keeping a *trusted certification path* from the PC to the CA that originally issued the user certificate.

A Proxy Certificate is an X.509 public key certificate with the following properties:

- It is signed by either an X.509 End Entity Certificate (EEC), or by another Proxy Certificate (PC). This EEC or PC is referred to as the “Proxy Issuer” (PI).
- It can sign only another PC. It cannot sign an EEC.
- It has its own public and private key pair, distinct from any other EEC or PC.
- It has an identity derived from the identity of the EEC that signed the PC. When a PC is used for authentication, it may inherit rights of the EEC that signed the PC, subject to the restrictions that are placed on that PC by the EEC.
- Although its identity is derived from the EEC's identity, it is also unique. This allows this identity to be used for authorization as an independent identity from the identity of the issuing EEC.
- It contains a new X.509 extension to identify it as a PC and to place policies on the use of the PC. This new extension, along with other X.509 fields and extensions are used to enable proper path validation and use of the PC.
- Although a PC also contains a validity period, they are quite short lived in comparison with the EEC. Formally speaking, a PC should live only for the duration of the task that was delegated to execute into the Grid.

The process of creating a PC is as follows:

1. A new public and private key pair is generated.
2. That key pair is used to create a request for a Proxy Certificate that conforms to the profile described in [RFC3820].
3. A Proxy Certificate, signed by the private key of the EEC or by another PC, is created in response to the request. During this process, the PC request is verified to ensure that the requested PC is valid (e.g., it is not an EEC, the PC fields are appropriately set, etc).

When a PC is created as part of a delegation from entity A to entity B, this process is modified by performing steps #1 and #2 within entity B, then passing the PC request from entity B to entity A over an authenticated, integrity checked channel, then entity A

performs step #3 and passes the PC back to entity B. We can imagine the procedure just described like a special case of cross-certification for Proxy Certificates.
Path validation of a PC is very similar to its analogous for End Entity Certificates, with a few additional checks to ensure, for example, proper PC signing constraints.
Using Proxy Certificates to perform delegation has several features that make it attractive:

- Ease of integration:
    - Because a PC requires only a minimal change to path validation, it is very easy to incorporate support for Proxy Certificates into existing X.509 based software. For example, SSL/TLS requires no protocol changes to support authentication using a PC. Further, an SSL/TLS implementation requires only minor changes to support PC path validation, and to retrieve the authenticated subject of the signing EEC instead of the subject of the PC for authorization purposes.
    - Many existing authorization systems use the X.509 subject name as the basis for access control. Despite its obvious disadvantages (there are no rules controlling the use of namespaces per Certification Authority), Proxy Certificates can be used with such authorization systems without modification, since such a PC inherits its name and rights from the EEC that signed it and the EEC name can be used in place of the PC name for authorization decisions.
- Ease of use:
    - Using PC for single sign-on helps make X.509 PKI authentication easier to use, by allowing users to "login" once and then perform various operations securely.
    - For many users, properly managing their own EEC private key is a nuisance at best, and a security risk at worst. One option easily enabled with a PC is to manage the EEC private keys and certificates in a centrally managed repository. When a user needs a PKI credential, the user can login to the repository using name/password, one time password, etc. Then the repository can delegate a PC to the user with proxy rights, but continue to protect the EEC private key in the repository.
- Protection of private keys:
    - By using the remote delegation approach outlined above, entity A can delegate a PC to entity B, without entity B ever seeing the private key of entity A, and without entity A ever seeing the private key of the newly delegated PC held by entity B. In other words, private keys never need to be shared or communicated by the entities participating in a delegation of a PC.
    - When implementing single sign-on, using a PC helps protect the private key of the EEC, because it minimizes the exposure and use of that private key. For example, when an EEC private key is password protected on disk, the password and unencrypted private key need only be available during the creation of the PC. That PC can then be used for the remainder of its valid lifetime, without requiring access to the EEC password or private key. Similarly, when the EEC private key lives on a smartcard, the smartcard need only be present in the machine during the creation of the PC.
- Limiting consequences of a compromised key:
    - When creating a PC, the PI can limit the validity period of the PC, the depth of the PC path that can be created by that PC, and key usage of the PC and

its descendents. Further, fine-grained policies can be carried by a PC to even further restrict the operations that can be performed using the PC. These restrictions permit the PI to limit damage that could be done by the bearer of the PC, either accidentally or maliciously.

- A compromised PC private key does NOT compromise the EEC private key. This makes a short term, or an otherwise restricted PC attractive for day-to-day use, since a compromised PC does not require the user to go through the usually cumbersome and time consuming process of requesting the CA to reissue another EEC with a new private key.

Next we describe an authorization framework used by most Grid installations, which is mostly based on the features delivered by the underlying GSI system just explained.

1.3.4. Grid Authorization Framework

The framework described in [AuthZFwk] is based on a more generic one previously introduced in [RFC2094], but in this case aimed to the Grid community although its concepts and ideas can be extended to other distributed systems and service oriented architectures (i.e. Web Services). The authors introduce the basic concepts and models of authorization, specifying a conceptual framework for Grids and classifying state-of-the-art and proposed authorization mechanisms with regard to it. Also this document considers the various authorization concepts and their relationships, describing a framework that is expected to be general and abstract enough to allow for the classification of any type of authorization system. Abstract entities and functions are defined and fundamental communication sequences for authorization requests and decisions are shown. As mentioned in the previous section, despite this framework does not consider authentication issues explicitly, they are highlighted as an essential element for the authorization process.

Authorization decisions are made based on information provided by authorities that must have a direct or a delegated relationship with either the authorization subject or with the resource that is the target of the request that triggered the authorization process, or even with both of them. These relationships may be implemented using a trust mechanism based on some cryptographic method or may be implemented completely off-line.

There are then three basic high-level entities involved in the Grid authorization process depicted in this framework:

- Subject: An entity that can request, receive, own, transfer, present or delegate an electronic authorization as to exercise a certain right.
- Resource: A component of the system that provides or hosts services and may enforce access to these based on a set of rules and policies defined by entities that are authoritative for the particular resource. Access to resources may be enforced by the resource owner or by some entity that is located between this resource and the requestor, thus protecting against unauthorized use.
- Authority: An administrative entity that is capable of and authoritative for issuing, validating and revoking by electronic means any proof such that the named subject of the issued electronic means is authorized to exercise a certain right or assert a certain attribute. Right(s) may be implicitly or explicitly present in the electronic proof. A set of policies may determine how authorizations are issued, verified, etc. based on the contractual relationships the Authority has established. Currently three types of authorities are in common use: Attribute Authorities, Policy Authorities and Identity Authorities.

The component performing the evaluation of the executable policy by computing an authorization decision on behalf of the authorities is sometimes referred to as an Authorization Server. Typically this entity may make or do a combination of: an authorization decision, an authorization lookup, and the delegation or proxy of an authorization decision to another AuthZ Server.

1.3.4.1.Authorization sequences

Entities introduced in this framework can be mapped into its correspondent principal according to [RFC2094] in the following manner:

- User → Subject.
- Service Provider → Resource.
- AuthZ Server → itself.

Considering this mapping the use cases for authorization sequences are described next.
The *Push Sequence* where the Subject first requests an authorization from an Authority. The Authority may or may not honor the Subject´s request. It then may issue and return some message or secured message that acts as a proof of right (called "authorization assertion"). Typically such an assertion has a validity time window associated with it. The assertion may subsequently be used by the Subject to request a specific service by contacting the Resource. The Resource will accept or reject the authorization assertion and will report this back to the requesting Subject.
Next is the *Pull Sequence*, consisting in the Subject contacting the Resource directly with a request. In order to admit or deny the service request, the Resource must contact its Authorization Authority, which will perform an authorization decision and return a message that obtains the result of this operation. The Resource will subsequently grant or deny the service to the Subject by returning a result message.
In the *Agent Sequence*, the Subject will contact a higher level agent with a request to obtain a service authorization. This agent will make an authorization decision based on the rules established by the authorization authority and if successful it will contact the Resource to provision a certain state as to enable the service. After receiving a successful feedback from the service, the agent will report success to the requesting Subject. This model is relevant to Grid users when requesting a certain QoS from the Grid system (i.e. resource reservation through a scheduler), as only then they may interact directly with the resource to access the service.
Finally is the *Hybrid Sequence*, which combines the basic sequences to cover particular authorization situations.

1.3.4.2.Domain considerations

An administrative domain defines the scope of authority. In Grid environments we frequently see scenarios where there are separate domains for identity, subject attributes, resource policy, and community policy authorities. For these Grid installations, a community or virtual organization (VO) domain is present. A VO domain can provide the Authorities that perform privilege management for all of its members. Grid Service Providers may also provide resources to users in multiple VO or home domains enhancing in this way the complexity of the authorization infrastructure required.
Almost in every case contractual relationships (often involving agreements) between the domains of the different Subjects, Authorities and Resources are frequently necessary to enable the acceptance and issuing of authorizations (trust relationships).

1.3.4.3.Authorization policies and Attributes

Authorization information such as policies, attributes, identities and environmental parameters are utilized and combined when making authorization decisions. Many policies use the concepts of "conditions" and "actions" which have to be evaluated with respect to the actual request, the requesting subject's identity and the attributes this subject holds.

Authorization attributes are statements about properties bound to an entity that implicitly or explicitly define its allowed actions on some resource. Attributes can be grouped into descriptive and privilege attributes. Descriptive attributes associate a feature with an entity, while privilege attributes define directly applicable access rights of an entity with respect to a resource.

1.3.4.4.Authorization Architecture

The framework explained in this section considers an authorization architecture consisting of a set of entities and functional components that allow authorization decision to be made and enforced based on attributes, parameters and policies that define authorization conditions. Figure 1 shows a Grid authorization architecture based on the pull model, even though similar structures can be generated for the push, agent and hybrid scenarios. This architecture will be used to explain some important details of the present framework, some of which are referenced along this thesis.

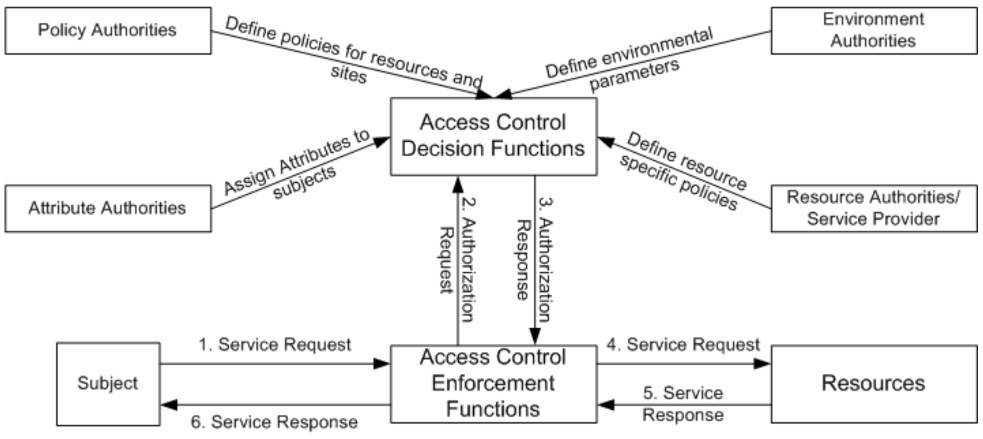

Policy Authorities
Define policies for resources and sites
Environment Authorities
Define environmental parameters
Access Control Decision Functions
Attribute Authorities
Assign Attributes to subjects
Define resource specific policies
Resource Authorities/ Service Provider
2. Authorization Request
3. Authorization Response
Subject
1. Service Request
6. Service Response
Access Control Enforcement Functions
4. Service Request
5. Service Response
Resources

Fig. 1. Authorization Architecture based on the pull sequence.

This framework uses the following two access control functions:

- Access Control Decision Function (ADF): Makes authorization decisions about a subject's access to a service. It is equivalent to the Policy Decision Point (PDP) defined in [RFC2904]. It is normally part of an Authorization Server and is independent of the Resource or Application, but it may be co-located with the Access Control Enforcement Function.
- Access Control Enforcement Function (AEF): Mediates access to a resource or service. It is equivalent to the Policy Enforcement Point (PEP) defined in [RFC2904]. It is often either integrated into or located in front of the Resource it protects.

Obviously the ADF, AEF, Subject and Resources may be embedded inside one or more administrative domains in a variety of combinations. In any case, there are three categories of information that may need to be passed between the subject, resource and various attribute authorities:

- Attributes: The subject may get the attributes from the attribute authority, the authority could pass the attributes directly to the relevant ADF, or it could put them in an attribute repository. When the ADF needs an attribute to make and authorization decision, it may get it from the subject either as part of the original request, during a negotiation phase or even may pull it from either a local repository or a repository associated with the attribute authority, the subject or virtual organization. Security considerations about attributes and its management (i.e. security controls) should be taken into account.
- Policy flow: Policies are typically stored in a repository or provisioned directly to the decision functions by the policy authorities. The transferring of policies has similar requirements to the transferring of attributes as it is imperative to securely establish the authority of the issuer and to protect the integrity of a policy.
- Authorization queries and responses: Authorization requests and responses are similar to attributes in that it is necessary to provide a secure binding. An authorization request must be securely bound to a subject and the subject's service request; on the other hand, the response must be securely bound to a request and when required also to its originator.

Evaluating a policy is the heart of the authorization decision process. The components responsible for expressing, storing, retrieving and evaluating policies can be thought of as a policy subsystem.
Policy expression is usually done by a policy language which contains the vocabularies to express various policy artifacts. If the policy language is extensible, then domain specific vocabularies and characteristics can be incorporated in the future without requiring fundamental changes. Access policy for resources is written by policy authorities that generally get their authority from the owner of the resource. The policy processing engines are likely to be proprietary to the various systems.
Some authorization decision may be indeterminate because there are conditions in the policy that the ADF cannot evaluate, involving the current state of the resource. In this case the conditions may be passed back to the AEF to evaluate, thus requiring from it to understand the policy language that is used to express these conditions.
Policy repository is also important to consider because once written the policies, they must be stored for use by an ADF. For security and efficiency reasons in most of the cases it is collocated with the ADF.
Finally, to exchange policies a substrate consisting of messages and protocols is required. Similar security considerations as in the case of attributes flow are required for policies transport.

1.3.4.5. Management Issues

In this framework trust management defines the authorities shown in figure 1 and specifies what they should be trusted to do. Policy and resource authorities both issue policies about resources, but the policy authority operates at a higher level and may issue access control policy for a whole site or VO. Attributes authorities assign attributes to subjects and may belong to the subject's domain or to a VO. Environmental authorities may define features about the resource environment (i.e. disk usage), or about the distributed environment (i.e. security of the connection).
Once a VO or resource domain knows how to represent various authorities, it needs to define which ones are to be trusted and for what purposes. The entity that defines these trust relationships is determined by the risk management strategy being used. The AEFs need to know exactly which ADFs to trust for authorization decisions.

A final item that comes under the category of trust management is the policy about who can create proxies which have all or some of the rights of the delegating entity, and who can delegate rights to other entities.
Privilege management covers the definition, assignment, storage, presentation, delegation and revocation of both privilege and descriptive attributes. For the management of privilege and descriptive attributes there are three distinct phases: granting the privilege, using the privilege, and removing the privilege. For privilege attributes there are two primary actors: the authority granting/removing the privilege, and the subject requesting-using the privilege. For descriptive attributes a third actor, called the policy authority, is required to associate authorization semantics in the access control policies with the descriptive attributes.
An attribute is granted to some entity by an authority for its relevant home domain. Sources of Authority –SOAs-, their delegates and the domains that will accept the attributes must have a common understanding of the authority's scope. This should be expressed in a privilege management policy, which among other features, defines which authority may grant what attributes, with what values and to what entities. An attribute authority may run on behalf of a number of authorities –delegation-, such as a real or a virtual organization. In most of the cases a method for querying the privilege management policy in order to determine the appropriate SOA for a given attribute is required.
Privilege management describes the process of defining who is allowed to which access rights. Privileges can be assigned by issuing a policy component describing direct access rights of a subject. The policy rule defining such a privilege typically consists of the three-tuple (subject, resource, action). If scalability is desirable then the presence of a descriptive subject attribute may be required (i.e. roles or group membership); in this case the policy rule consists of a three-tuple (attribute, resource, action) and the attribute is a two-tuple (subject, attribute). Hierarchical role assignment extend this concept even more by allowing for access right inheritance from less privileged to more privileged roles. As part of privilege management also shall be considered the revocation of such roles.
Attribute assertions are proofs of the right to assert a descriptive attribute or privilege attribute. The list of attributes one has is generated from the list of attribute assertions one possesses and this may allow the subject to act as a SOA for those attributes within its own domain (even allowing delegation as described by the correspondent attribute authority). These assertions must be stored pending their use in the so-called "attribute repository", which may need special security considerations as mentioned in previous sections. As attributes must be understood by relying parties in often multiple administrative domains (following the Grid's philosophy), a common schema for its syntax is required. In any case the attribute authority may choose to issue attributes in a variety of ways, but it is recommended to keep them separated from identity tokens (i.e. identity certificates). Finally, these attributes must be asserted to the ADF by some protocol (special considerations may arise in the case of distributed attribute management).
Policy management for static resources is commonly solved by the use of a repository. However creating a policy for dynamically created objects is even more challenging and in many cases is preferable to control access to a whole class of objects by a static policy and then just create the new ones within these classes.
Some of the issues that are addressed by policy management are: who can create, modify and delete policies for each resource; also important are: how quickly policies can be revoked and where does the ADF finds them -considering trust relations-. In

many cases an additional challenge is raised with distributed policy management (i.e. how to find all the relevant pieces to take a decision, displaying the current policy to the resource owner or to anyone trying to add to the policy). State of the art technologies reviewed in section 2.1 will show several approaches to solve this problem.

1.3.4.6. Authorization context

This consists of those properties of the Authorization Request which are neither provided via Authorization Attributes nor included in Authorization Policies, but which are relevant to the decisions made by the Authorization Server. The Authorization Context may include environment information (i.e. disk usage) and authentication information (i.e. key lengths, etc.).

The authorization context and its relationship with authentication and authorization parameters will be important for the establishment of the proposal presented later in this thesis.

1.3.4.7.Authorization server

The Authorization Server is an entity that evaluates authorization requests and issues responses, taking into account relevant attributes, policies and environmental parameters.

Commonly the authorization algorithm used takes all or some of the following inputs:

- Nature of request.
- Attributes of requestor.
- Attributes of resources.
- Authorization context.
- Environmental factors.
- Policy statements.

Authorization responses are typically the output from these algorithms and may be specified in a particular assertion language.

1.3.4.8.Enforcement mechanisms

This work is done by limiting the delegated operations performed on resources -on behalf of a subject- to those permitted by an authoritative entity. A common practice on these mechanisms focuses only on user's access to an application. However this is not completely certain in Grid environments, where applications and services can be provided by software components that are staged by the user to the computer resource and are not necessarily trusted by the resource owners. The resources act as hosting environments for these services, which often are transient and migrating between different resources to meet a certain performance criteria. In other words, it is important to control not only the access of a subject to a service but also the access of a service to its current service-hosting environment.

Enforcement functions can either receive the set of authorized operations as part of the service request, or also by querying an ADF (it depends on the message sequence being used). In any case, enforcement mechanisms can be characterized in two different groups:

- Application dependent enforcement mechanisms: these are often integrated into the application or service and perform enforcement functions before attempting to access underlying operating system resources.
- Application independent enforcement mechanisms: are separated from the service or application and take the approach of running the service in a very constrained execution environment (maybe being enforced by the kernel or by intercepting the respective call before being passed to the operating system).

1.4 Motivation: bridging the interoperability gap through Validation

Current Grid Authentication and Authorization (AA) frameworks are capable of delivering a certain level of assurance to resource owners in relation to a Grid user, but lack of important features required to build *Grid interoperability* among untrusted domains. When dealing with users belonging to well-known domains, these AA mechanisms are enough to provide a *minimum trust level* related to the identity assertions contained in the authentication credentials; for example in the case of cryptographic X.509 digital certificates, the resource owner will implicitly trust in the underlying registration process performed by the respective Certificate Authority. If a minimum level of confidence or *trust* can not be given to these identity credentials, then subsequent authorization processes can not proceed.

Nowadays however, we have seen in this chapter that Grid technology is beginning to expand out of the traditional academic field where it was created and it is starting to reach new frontiers: industrial, financial, eHealth, etc. In these scenarios there are so many new variables being introduced that explicit trust assumptions are no longer valid: researchers belonging to different Institutes -under possibly unknown Certification Authorities' domains- that wish to interoperate, users from private companies with certificates issues by complex PKI hierarchies, Virtual Organizations dynamically being created and removed, etc. Under these new scenarios Grid authentication and authorization mechanisms have been unable to convey enough information to build *trust* relationships among the different administrative domains that usually appear into the Grid.

A first approach took by this research to *bridge this interoperability gap*, was the *Unified AAI framework for Grid Services*, which considered the following elements:

- A Conceptual Framework from researching Grid Services' features related to Authentication and Authorization that may be unified (opposed to the traditional vision of treat them as independent features), in such a way that important security and performance issues may be obtained.
- An Accreditation Policy that implemented such Unified Framework through a set of rules about Authentication and Authorization.
- A Trust Engine system that evaluated the distributed accreditation policy.
- An AAI architecture for the Globus Toolkit built upon the Unified Framework, Accreditation Policy and the Trust Engine concepts mentioned above.

However after a few months, it was clear from our initial work within other research communities like the Open Grid Forum that more important than *unifying* the underlying authentication and authorization processes, it was necessary to *bridge* them together so only *trusted* and *interoperable* credentials could be conveyed from the authentication to the authorization subsystem. In this way our research focused on the properties required to build the mentioned "bridge", which resulted finally into the concept of *Grid Validation* just as explained next.

1.5 Grid Validation

One question common to current Grid scenarios is: How can Grid relying-parties trust each other in order to interoperate? More specifically, is it possible to quantitatively measure the trust level associated with a Grid identity at a certain moment? Both of these questions are related with a concept that, despite it is not new to other distributed systems, it is just being introduced into Grid environments: *validation of end-entities*.

In the context of the present thesis, *Validation* is defined as the process that, staying between the Authentication and Authorization processes, asserts that a cryptographic credential is related with a specific *trust level*. For example, validating a Proxy Certificate -the most widely used X.509 credential in nowadays Grid installations-, is an iterative process that involves performing a cryptographic verification on its digital signature, checking its validity period and asserting that it has not been revoked previously. This process is known as *Basic Path Validation* [RFC3820].
Despite the fact that Basic Path Validation represents a step forward towards improving Authentication and Authorization processes, it is still quite limited when a Grid relying party requires to associate an end-entity credential or identity assertion with a particular trust level, for example in order to allow an specific subset of actions on a Grid resource.
In the present thesis, the process that associates a Grid Identity (i.e. an X.509 end-entity certificate) with a specific trust level is called *Extended Path Validation*.
Any Grid security system implementing the Extended Path Validation process must at the very least supply to the relying parties:

- A near-real time mechanism to provide up-to-date status information about an end-entity certificate, and
- A formal methodology to evaluate the trust level associated with this end-entity certificate.

This thesis proposes the use of a Grid Validation Infrastructure that implements the Extended Path Validation procedure for Proxy Certificates to provide interoperability among unrelated Grid-PKIs. The next section briefly describes the main components of this validation system.

1.6 Originality and contributions

The contributions presented in this thesis can be classified in two classes. In first place a methodological approach for the Extended Path Validation in Grid environments, and in second place an approach that proposes an architecture for implementing a Grid Validation Infrastructure. Sections 1.6.1 and 1.6.2 briefly introduce the particular contributions offered by both approaches.

1.6.1. Methodological contributions

This thesis presents the following contributions to the Grid validation arena:

- *Grid-OCSP:* Despite the fact that the Online Certificate Status Protocol (OCSP) was released in 1999, its use in Grid environments has just begun to be studied a couple of year ago. This may explain why most of our research represented an original achievement, especially to the Open Grid Forum where we have been actively working in the development of an open recommendation for the Grid Community. An article about our Grid-OCSP proposal achieved a best paper award in an international Grid conference, and also was published in an international journal.
- *Static Policy evaluation:* The second goal of our research on Grid validation was achieved in a joint research with Neapolitan researchers by extending the Reference Evaluation Methodology (REM) to be used in Grid environments. When the first phase of this research finished, the Grid-PKI static evaluation technique proved useful to aid Grid Trusted Third Parties (usually called Policy Management Authorities or PMAs) in their accreditation process: evaluating if a Grid Certificate Authority fulfills the minimum security requirements of the PMA, so it may become a member and thus could interoperate with Grid users from other accredited CAs. At

the time of writing this thesis, the static policy evaluation technique was about to be presented to the Grid community in an international conference so we expect it to be adopted into the PMA's "toolbox" in the near future.

- *Dynamic Policy Evaluation:* This milestone represents the fusion of both the Grid-OCSP and the static policy evaluation, to implement the Extended Path Validation algorithm for Proxy Certificates. The approach is simple (use OCSP to convey evaluation data), but quite useful to help entities from different CAs to dynamically decide if they can interoperate into a new Grid Virtual Organization. Comments from the Grid community have been positive about the usefulness of our approach, and also a paper was published in an international conference about this topic.

1.6.2. Architectural contributions

The methodological research presented in this thesis has been complemented with an architecture proposed to implement a Grid Validation System with the following elements:

- The Open GRid Ocsp (OGRO) API, an open source prototype that along with the project CertiVeR formed –at our best knowledge- the first Grid-OCSP Validation Infrastructure.
- A module implementing a formal technique extended from the "Reference Evaluation Methodology (REM)" (originally proposed by researchers of the "Università di Napoli, Federico II" and the "Seconda Università di Napoli"), which can be applied to Grid-PKIs to *i)* determine if an end-entity certificate is compliant with the minimum security requirements defined by the relying party, and to *ii)* define a security metric that quantitatively computes a Global Security Level (GSL) representing how much better the end-entity's Certification Policy security provisions are when compared against the relying party's requirements.
- The Policy and OCSP based Interoperability System (POIS), an infrastructure able not only to convey the status and associated GSL of any end-entity's Proxy Certificate in almost real-time (via OCSP or SOAP), but also designed to solve the special challenges posed by the computing and the data Grid.

1.7 Thesis organization

The rest of thesis work has been structured in chapters ordered according to the milestones of our research. Each one of these chapters denotes in some degree an original contribution made to the Grid-security community, and in most of the cases this is illustrated by the correspondent publications or developed prototype.

In *Chapter 2* is presented the State of the Art related with Validation and Grid Services; this will clearly show the open issues found with current technologies.

Chapter 3 presents our Grid-OCSP proposal, a methodological contribution towards using OCSP as a validation mechanism for the Grid.

In *Chapter 4* is explained our approach for extending a security evaluation technique to accredit Grid PKIs via a trusted party (Policy Management Authority).

Chapter 5 describes the methodology proposed for implementing an Extended Path Validation mechanism, which is able to dynamically provide interoperability to the Grid.

The details behind the implementation, deployment and performance of the proposed Interoperability System are presented in *Chapter 6*.

Finally conclusions and future work are presented in *Chapter 7*, while published articles are referenced in Appendix B.

CHAPTER
2 STATE OF THE ART

2.1 Grid Authentication and Authorization

During the last years, security problems related with authentication and authorization on distributed systems have been widely studied and also an equal number of AAI systems have been deployed in production environments. New problems related with interoperability and scalability issues arose when AAIs entered into the Grid arena, where also existed a need for reusing legacy technologies to reduce the total cost of ownership.

This section reviews a set of widely deployed Authentication and Authorization Infrastructures, with a particular emphasis in Computational and Data Grid environments.

2.1.1. Virtual Organization Membership Service (VOMS)

VOMS [VOMS] is an Attribute Authority that exposes attributes and encodes the position of the holder inside the Virtual Organization (VO). VOMS uses X.509 identity and proxy certificates and it is being extended to support SAML protocol and assertions as defined in [SAML].

Nowadays VOMS is a widely-used attribute authority service and also the de-facto AAI in most European production Grids.

VOMS works according to the push authorization model: the user, before contacting the Grid service, will contact a VOMS server to obtain his corresponding Attribute Certificates (ACs) [RFC3281] and then convey them to the target Grid service using conventional X.509 proxy certificates compliant with [RFC3820]. These operations are depicted in figure 2.

A VOMS holder may be a member of several groups, and may hold a special role inside some of his groups. Groups are organized in a tree structure that comprises groups and subgroups, while roles are not hierarchical and therefore are associated to the group membership. Worth to notice is that the internal structure of a VO, as defined in VOMS, is not a full hierarchical Role Based Access Control (RBAC).

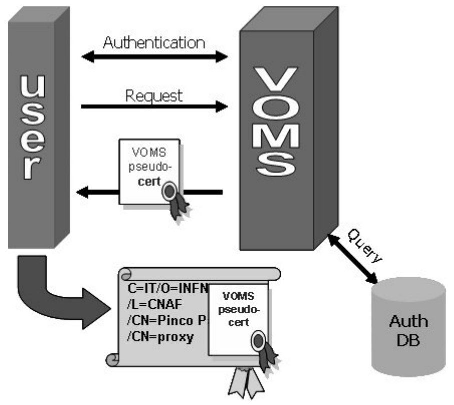


Fig. 2. The VOMS Authorization model.

2.1.2. Community Authorization Service (CAS)

The CAS authorization service [CAS] is built on top of the underlying GSI subsystem. A user requesting access to a resource contacts the CAS server which, after a GSI-based authentication, issues a restricted proxy certificate with an embedded access-control policy. The user utilizes this credential to connect to the requested resource, and then the resource itself applies its local policy to further restrict this access. CAS does not record groups or roles, but only permissions. A schematic view over CAS' policy management and enforcement is shown in figure 3.

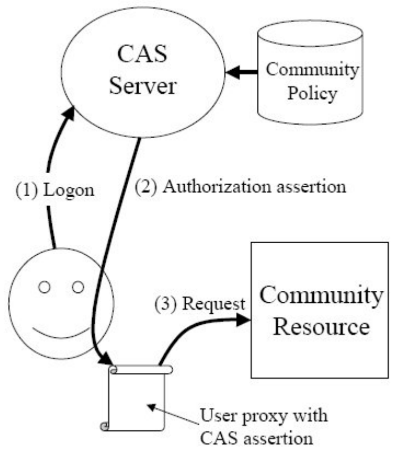


Fig. 3. The CAS Authorization model.

CAS allows a VO to maintain its own set of policies explicitly and communicate those policies to sites. The sites then combine their local policies (about what the VO is allowed to do) with the VO's policies (about what the individual user is allowed to do as a VO member) and enforce this combined policy.
The VO, through administration of the CAS server, maintains the VO's portion of this combined policy. This portion of the policy includes the following:

- The VO's access control policies regarding its resources: which rights are granted to which users (e.g., which users can read which files);
- The CAS server's own access control policies, such as who can delegate rights or maintain groups with the VO. These policies can be expressed at a fine grained level (e.g., a user may be allowed to grant rights on only certain resources, or add users to only certain groups);
- The list of VO members.

The other part of this combined policy, the resource provider's policy regarding the VO, is maintained by the resource provider himself by using the same native mechanisms implemented for non-VO users. For example, a site may create a local identity representing a VO and add local configuration mapping users presenting credentials from that VO's CAS server to that identity. The site would then use local mechanisms to set policy on the VO as whole, for example, change file ownerships to allow the VO identity read and write access to a particular subset of the file system, or set file system quotas limiting the amount of space that the VO can use. A resource provider may use this mechanism to maintain policies for several VOs, each running its own CAS server.
Resource providers participating in a VO with CAS will deploy CAS-enabled services (i.e. services modified to enforce the policy in the CAS credentials) onto resources they assign to the VO. A user wishing to access those resources first contacts the VO's CAS server and requests a CAS credential. The CAS server replies with a CAS credential that contains a policy statement of that user's rights, cryptographically signed by the CAS server.
When making a request to the resource, the user presents the CAS credential. Upon receiving the CAS credential, a CAS-enabled service takes several steps to enforce both VO and local policy:

- Verify the validity of the CAS credentials (e.g., signature, time period).
- Enforce the site's policies regarding the VO, using essentially the same method as an unmodified server. However, the identity used when enforcing the site's policies is the identity of the signer of the policy assertion (i.e., the VO's CAS server), not the identity of the individual user authenticating.
- Enforce the VO's policies regarding the user, as expressed in the signed policy statement in the CAS credential.
- Optionally, enforce any additional site policies in regard to the user (for example, a site may keep a blacklist of end users who are not allowed to perform any action, regardless of any VO policy).

2.1.3. PRIMA

PRIMA [PRIMA], is a system for PRIvilege Management and Authorization that provides enhanced grid security services. The requirements for these services are derived from usage scenarios and supported by a survey of grid users performed by the authors. These basic requirements for added flexibility, increased expressiveness, and more precise enforcement are met by a combination of three mechanisms:
Use of secure, fine-grained privileges representing externalized access rights for grid resources that can be freely created, shared, and employed by grid users;
A dynamic policy generated for each request combining the request's user-provided privileges with the resource's access control policy; and Dynamic execution environments specially provisioned for each request that are enforced by the resource's native operating system and which support legacy applications.
Figure 4 provides a high-level overview of the PRIMA system using numbered arrows to represent a general sequence of actions. In step 1, resource administrators configure grid resources with access policies that regulate usage of the resource. These policies can be created and managed at the grid layer using platform independent languages such as XACML [XACML]. Access rights of resource objects are externalized and represented to both administrators and users on the grid layer as PRIMA privileges (step 2). For example, a privilege for a file access right is abstracted from the way file access rights are stored in the file meta information by the native operating system. The PRIMA representation uses XACML constructs to encode the externalized form of the privilege. A privilege is self-contained in that its meaning is fully determined by the information contained in the privilege. Step 3 shows that privileges can be traded among and between grid users, administrators, and other grid entities (such as grid services, proxies, agents, etc.). Grid users holding privileges can selectively use (step 4) and apply these privileges to grid resource requests (step 5).
Because the holder of privileges can selectively provide individual privileges to grid resources when requesting access, least privilege access to resources is enabled and the user is ensured of fine-grained control over resource usage of requested services. Privileges form a key element of PRIMA. Authorization decisions are made based on the resource's access policy and the privileges provided with the request. The combination of the user's privileges with the resource's security policy prior to the assessment of the user's request is termed a "dynamic policy". Finally, the request is executed in an environment configured with the minimal set of fine-grained access rights required for the specific access. Enforcement via the execution environment provides for the secure execution of non-trusted legacy applications without the need to duplicate security code already present in the operating systems.

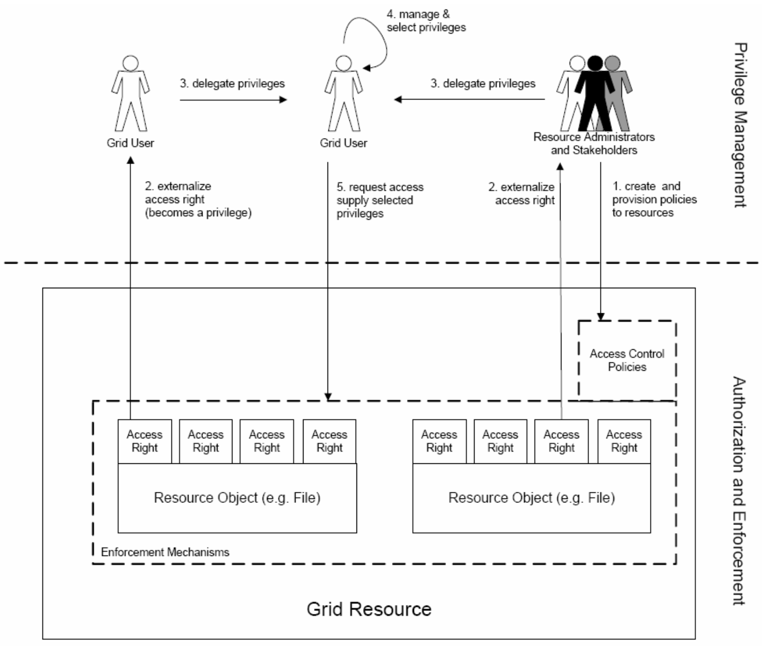

4. manage & select privileges
3. delegate privileges
3. delegate privileges
Grid User
Grid User
Resource Administrators and Stakeholders
Privilege Management
2. externalize access right (becomes a privilege)
5. request access supply selected privileges
2. externalize access right
1. create and provision policies to resources
Access Control Policies
Access Right
Resource Object (e.g. File)
Enforcement Mechanisms
Grid Resource
Authorization and Enforcement

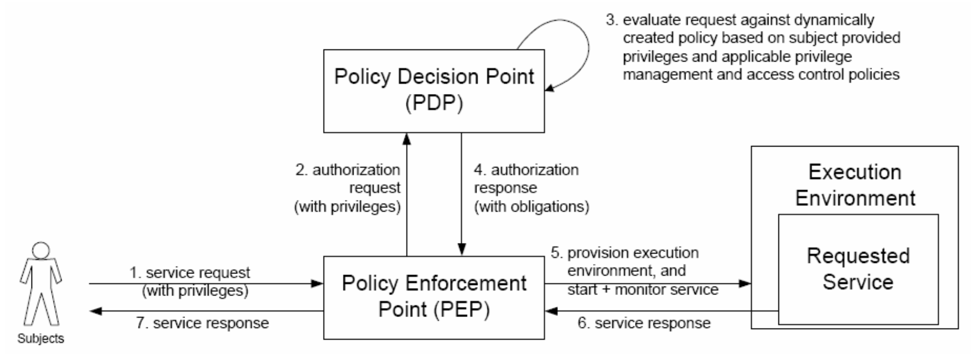
Fig. 4. An overview of the PRIMA system.

When a subject issues a service request he also provides a set of privileges. The request and the accompanying privileges are presented to a Policy Enforcement Point (PEP). For each provided privilege, the enforcement point individually checks the:

- Applicability (does the privilege apply to the local resource and the entity requesting the service),
- Validity (is the privilege within its lifetime and is the digital signature intact), and
- Authority (was the privilege issued by an authoritative party).

Authority to issue privileges is defined in a privilege management policy available to the Policy Decision Point (PDP). The PEP queries the PDP for authoritativeness of each privilege issuer before further considering a provided privilege. All permissible privileges constitute the dynamic policy for the request. The enforcement point can now contact the decision point with an authorization request which includes the dynamic policy. The PDP evaluates the request against the combination of all applicable policies (including the dynamic policy) and provides an authorization response back to the enforcement point.

Conceptually two decisions are made by the PDP: (1) a relatively coarse decision corresponding to the service request that specifies if the Policy Enforcement Point (PEP) should allow access to the requested service in general; and (2) a set of instructions, termed obligations, from the PDP to the PEP on how the requested service should be confined and monitored during its execution. If the PEP cannot fulfill the obligations then it should not allow the access to proceed. The enforcement point, upon receiving a positive response from the decision point, instantiates a custom execution environment configured with the access rights as specified in the obligations, and starts and monitors the execution of the requested service in this environment. Finally service responses are passed to the enforcement point and relayed to the subject. The overall authorization model is shown in figure 5.


3. evaluate request against dynamically created policy based on subject provided privileges and applicable privilege management and access control policies
Policy Decision Point (PDP)
2. authorization request (with privileges)
4. authorization response (with obligations)
Execution Environment
Requested Service
5. provision execution environment, and start + monitor service
6. service response
Policy Enforcement Point (PEP)
1. service request (with privileges)
7. service response
Subjects
Fig. 5. The PRIMA Authorization model.

2.1.4. Akenti

Akenti [Akenti] is an authorization infrastructure from the Lawrence Berkeley National Laboratory in the USA, which can be considered an AAI as most of its security is build around the previous authentication process. It represents the authorization policy for a resource as a set of (possibly) distributed certificates digitally signed by unrelated stakeholders from different domains. The policy certificates are independently created by authorized stakeholders. When an authorization decision needs to be made, the Akenti policy engine gathers up all the relevant certificates for the user and then the resource verifies them, determining the user's rights with respect to it.
The Akenti model consists of resources that are being accessed via a resource gateway (PEP) by users. These users connect to the resource gateway using the SSL handshake protocol to present authenticated X.509 certificates. The stakeholders for the resource express access constraints on the resources as a set of signed certificates, a few of which are self-signed and might be stored locally or remotely, but always in a secure way. These certificates express the attributes a user must have in order to get specific rights to a resource (called "Use-condition certificates"), who is trusted to create use-condition statements and who can attest to a user's attributes (called "Policy Certificates"). At the time of the resource access, the resource gatekeeper (PEP) asks a trusted Akenti server (PDP), what access the user has to the resource. The Akenti server finds all the relevant certificates, verifies that each one is signed by an acceptable issuer, evaluates them, and returns the allowed access. This authorization model can be seen in figure 6.
Akenti policies [AkPol] and [AkCom] are distributed and hierarchical. Each policy compromises two components: Use-condition certificates and Policy Certificates. The former comprise a root policy certificate, and optionally subordinate policy certificates that inherit from the root policy. Akenti sees the target as a tree of resources (like a filesystem with subdirectories). Each of the branches (subdirectories) can have a policy of its own, but in addition to that the policy of the superior branch (directory) is inherited. A stakeholder is a special kind of authority that is trusted to issue Use-Condition certificates and digitally sign Policy Certificates. Each stakeholder can impose his own access control requirements independently of other stakeholders.
Use-Condition certificates contain the name of the target resource, a condition (which can be a constraint), a critical flag, the authorities of the certificates with the attributes to be matched against these conditions, plus a list of rights/privileges that are granted.

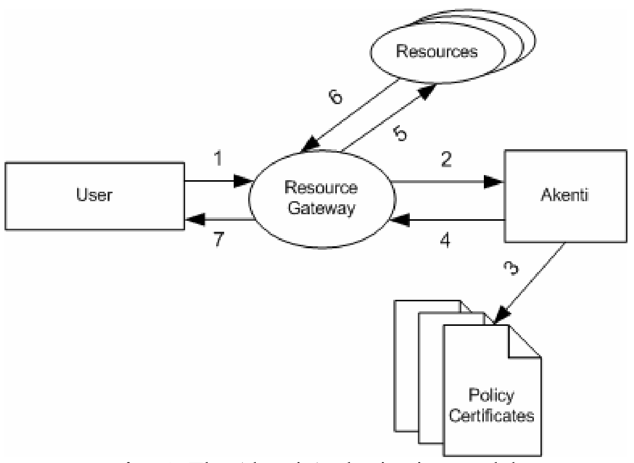


Fig. 6. The Akenti Authorization model.

Conditions may include identity attributes that the users must have, role or group membership and environmental parameters. Rights are comma separated lists of actions on targets. Action names have to be unique for the whole domain of resources, irrespective of the target type. The authority trusted to issue each attribute value must be specified exactly, but each Use-Condition can include different authorities for each attribute. In Akenti conditions are placed on which attributes certificates can be trusted and on which attributes have which privileges on a certain moment of time (in the Use-Condition certificates). The attributes issued to users in attribute certificates are independent of each other, and cannot form a role hierarchy. Akenti supports the distributed management of attribute certificates and an external Attribute Authority may assign attributes to users if the Use-Condition certificate lists it under the relevant attribute value.

Akenti's Policy Certificates consists of:

- Name of the resource to which this policy applies.
- List of information about trusted CAs including:
  - Distinguished name.
  - Public key certificate (can be self signed).
  - List of places to search for identity certificates issues by this CA (optional).
  - List of locations where CA stores its CRLs (optional).
- List of Use-Condition issuers (defines the resource stakeholders).
- List of URLs to search for Use-Condition certificates.
- Optionally URLs to search for user attribute certificates.
- Maximum time in seconds that any certificates that are used in satisfying conditions for this resource may be cached.

All Akenti credentials are built in a proprietary format using XML syntax.

In Akenti, Decision making is done in a single step, but the system is able to make different types of decisions. The client may ask "What can a user do?", as well as the traditional "Can this user perform this action on this target?".

Akenti always embeds its authorization responses in a Capability Certificate for export back to the client. The Capability Certificate comprises the public key of the user, his DN and his CA's DN, the name of the resource, and the privileges that the user enjoys with an optional list of conditions attached to each of them. The Capability Certificate is then signed by Akenti and given to the client. The user can subsequently present this to a gatekeeper for improved performance. The gatekeeper, which holds the Akenti public key, merely needs to check the signature on the capability, then ask the user to sign a

challenge, before granting (or denying) the user access to the resource. Trust chains in Akenti begin at the root policy, which is signed by a trusted stakeholder and must be securely configured into Akenti itself.
Akenti is written in C++ and its compliance checker can be called either via a function call in the gateway or as a standalone server via TCP/IP.
All related authentication processes are PKI-dependent and the user must present a valid X.509 public key certificate at authentication time, so the authorization process can be invoked later on.

2.1.5. PERMIS

PERMIS [Permis] is an authorization infrastructure from the European Comission funded "PrivilEge and Role Management Infrastructure Standards validation" (PERMIS) project. It implements a compliance checker that can be invoked as a Java object from a proprietary gateway. All the cryptographic credentials used in PERMIS are built according to the latest X.509 standard.
Being authentication agnostic, PERMIS leaves to the application the task of determining what type of mechanism should be used for this function. On the authorization side, policies are held into X.509 Attribute Certificates and PERMIS has implemented role based access controls –RBAC-.
The PERMIS architecture comprises a privilege allocation subsystem and a privilege verification subsystem (figure 7). There can be any number of privilege allocation subsystems in a fully deployed system. This subsystem issues X.509 role assignment attribute certificates to users and stores them in a LDAP directory for subsequent use by the privilege verification subsystem. In addition to privilege allocation, each user will also need to be issued with an application specific authentication token. If a PKI is being used, this token will be a digitally signed public key certificate, but if a conventional authentication system is being used, then it will be a username/password pair. Authorization is performed in an application-independent manner according to the PERMIS RBAC authorization policy. In this way one policy can control access to all resources in a domain.
The PERMIS policy is bound to a specific subject and is stored in an LDAP directory as a policy attribute certificate. It supports the classical hierarchical RBAC, in which roles are allocated to users and privileges to roles. Superior roles inherit the privileges of subordinate roles in the hierarchy. Multiple disjoint role hierarchies can be specified. PERMIS has a very loose definition of a role so it may be any attribute assigned to a user. The system also supports the distributed management of attribute certificates, and multiple external Sources of Authority –SOAs- can be trusted to issue roles/attributes.
PERMIS policies are kept in the LDAP entry of the policy issuer and different policies are distinguished by their unique object identifier (OIDs). There is no need for the policies to be kept securely, since PERMIS engine validates the policy at run time to ensure it is the correct one. However the name of the SOA has to be securely configured into the PERMIS application at start up. Even though policy hierarchies are not supported by PERMIS, they can be either enforced organizationally by management, or by the application instantiating several PERMIS decision engines, one per level of the hierarchy, and ensuring that each level grants permission.

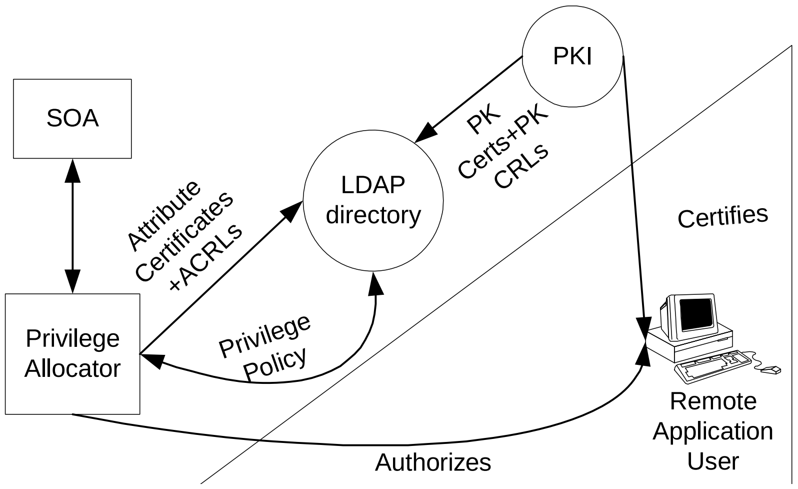


Fig. 7. PERMIS: privilege allocation subsystem.

The PERMIS policy is expressed in XML and their components comprise:

- Policy OID, so the policy can be distinguished among others stored in the SOA's entry.
- Subject domains, specified as LDAP subtrees, which are the subjects who can assert roles.
- Target domains, specified as LDAP subtrees, which are the targets governed by the Target Access Policy.
- List of the distinguished names of trusted external attribute certificate issuing authorities (SOAs), which are treated as roots of the delegation trees.
- Role hierarchy specification (list of the roles as ordered attribute values).
- A role assignment policy that tells which attribute authorities are trusted to issue which roles to which subject domains, and whether delegation is supported or not.
- Action policy, saying what the actions and their parameters are, so they can be referenced in the Target Access Policy.
- Target Access Policy, which specifies the set of roles/attributes required to perform a particular action along with any conditions.

PERMIS operates in multi-step decision making mode (figure 8). In step 1 with the function *getCreds*, the user's credentials are obtained and validated, and roles conforming to the policy are passed back to the calling application for caching. This typically takes place during user login. In step 2 the requested action and target are passed to the decision function along with the user's validated roles, and at the end a simple boolean decision is returned, either granting or denying access. As the user attempts to perform different tasks, the second step can be repeated as many times as required for different targets and different actions. In this model, a user accesses resources via an application gateway. The Access Control Enforcement Function –AEF- authenticates the user and then asks to the Access Control Decision Function -ADF- if the user is allowed to perform the requested action on the particular target resource. The ADF accesses one or more LDAP directories to retrieve the authorization policy and the role Attribute Certificates –ACs- for the user, and bases its decision on these pieces of information.

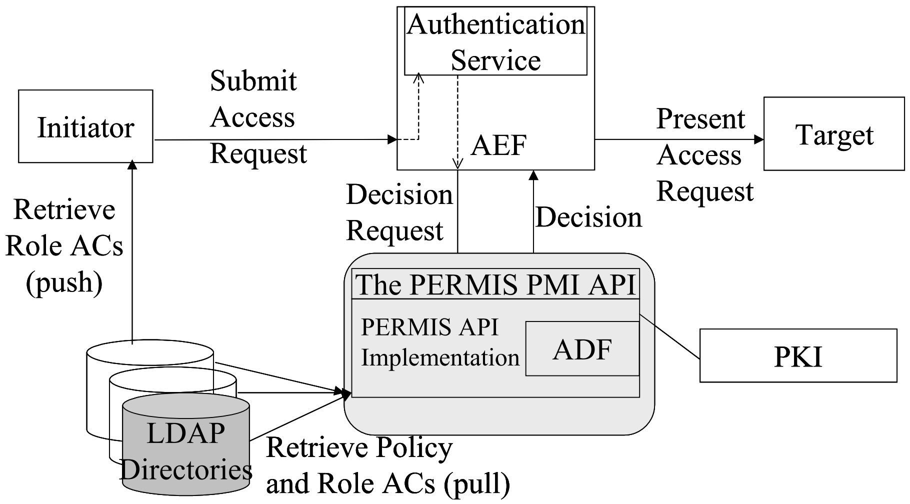


Fig. 8. The Privilege verification subsystem.

The PERMIS API has been written in Java and due to its structure (the Target and the AEF are co-located or can communicate with each other across a trusted LAN), the authorization token carried between them doest not need any protection at all. Also was assumed that only a single authorization service will be available at any time, and that the API does not need to support the export of authorization tokens, as attribute certificates are already in an exportable format.

2.1.6. Shibboleth

Shibboleth [Shib] is a joint project of Internet2/MACE (Middleware Architecture Committee for Education) and IBM. It aims to develop an architecture for standard-based and vendor-independent web access control infrastructure that can operate across institutional boundaries.

The focus of Shibboleth is on supporting inter-institutional authentication and authorization for access to web-based applications. The intent is to build upon existing heterogeneous security systems in use on campuses today, rather than mandating particular schemes like Kerberos or PKI based on X.509. The Shibboleth Project will produce an architectural analysis of the issues involved in providing such inter-institutional services, given current campus realities; it will also produce a pilot implementation to demonstrate the concepts.

In Shibboleth the origin site (where the client belongs) is responsible for authenticating the user, and for providing attribute information about the user to the target. The target site (where the resource manager belongs) is responsible for comparing these statements against the policy rules associated with the desired resource. Both sites will need to satisfy themselves as to the true origin of a request or a set of assertions. Every site that joins Shibboleth can decide to participate as an origin, as a target, or as both. In any case the joining institution must make available a PKI certificate containing the public key of their Shibboleth signing entity. All statements from the institution must be signed using the matching private key. When a site receives a signed request from another site, it must be easy for the recipient to securely obtain the requestor's certificate, in order to validate its signature.

Shibboleth uses pseudononymous identifiers in conversations between the origin and target sites, thus protecting user's privacy. In the case of Single Sign-On (SSO), Shibboleth does not provide such mechanisms but instead provides a way for intercampus SSO systems to interoperate between them.

According to [AACE-B1] the primary design principles for Shibboleth are:

- No single central piece of infrastructure is required, thus providing scalability.
- Data protection and privacy are of importance for Shibboleth.
- The User is guided by HTTP redirections from the Resource to the Authentication Server and back to the Resource for the authorization

Shibboleth uses a federated administration where a Resource Owner leaves the management of User identities and attributes to the User's Home Organization, which is also responsible for providing attributes about a User (possibly but not necessarily including a username) that the Resource Owner can use in making an access control decision when the User attempts to use its Resource. Users are registered only at their Home Organizations, and not at each Resource.

Shibboleth can be seen as a system for securely transferring User attributes from the User's Home Organization to the site of the Resource Owner, provided the Resources are accessible via standard web browsers. In addition, Shibboleth enables the Users to decide which information about them gets released to which site. The Users therefore have to balance access and privacy.

The major components of Shibboleth (shown in figure 9) are:

- Where Are You From Server (WAYF): Redirects the User to the HS at his/her Home Organization. At least one WAYF server is needed but it may be replicated as desired.
- Handle Server (HS): Authenticates a local User according to the methods of the Home Organization and provides an opaque handle identifying the User.
- Attribute Authority (AA): Retrieves the attributes, which a User allows to be given to a Resource (according to the User's Attribute Release Policy) and passes them to the SHAR on behalf of the Resource.
- Shibboleth Indexical Reference Establisher (SHIRE): Makes sure that the Resource gets a "pointer" (handle) back to the User without requiring more knowledge about it. In case this handler is missing, then it refers the User via the WAYF Server back to his/her HS to get one again.
- Shibboleth Attribute Requester (SHAR): Contacts the AA to fetch the available attributes describing the User and passes them on to RM.
- Resource Manager (RM): decides on access to the Resource based on the information received, and where necessary, in the information about sessions of the same User.

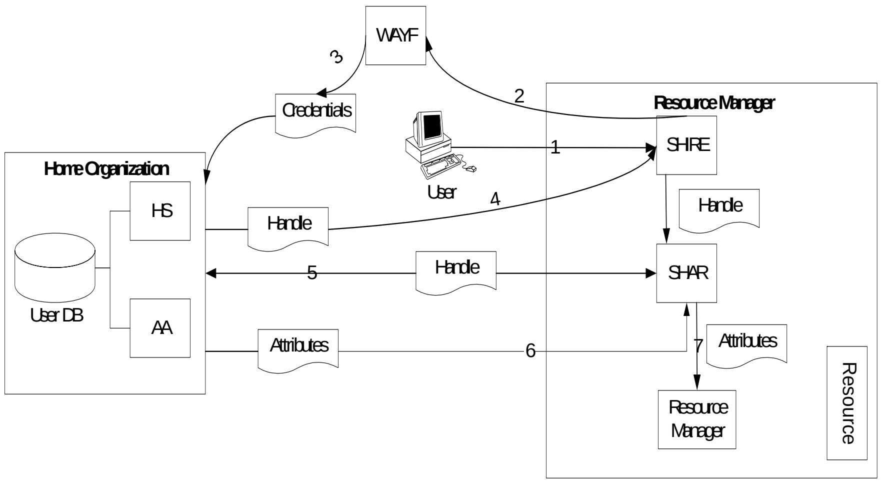
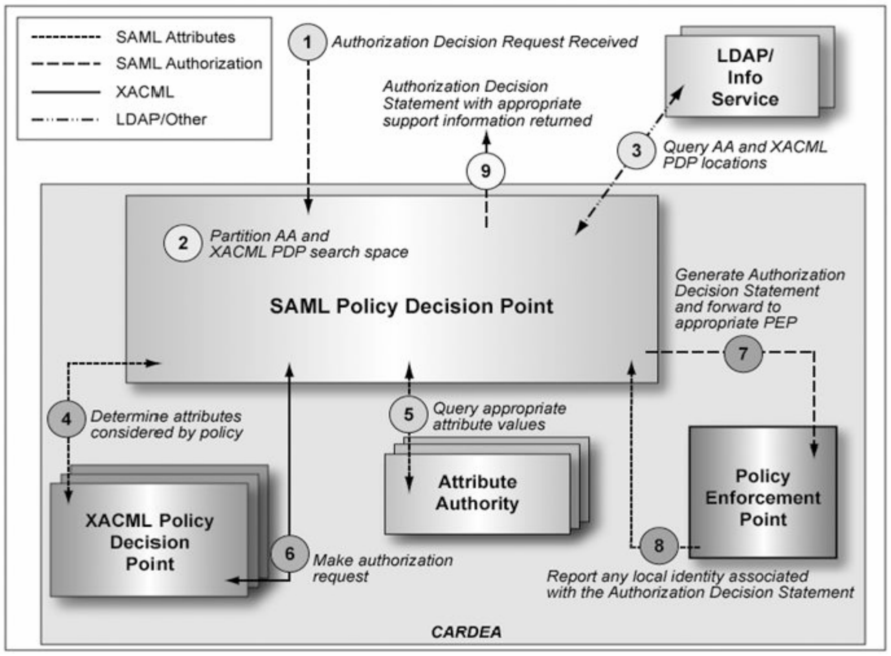


Fig. 9. Shibboleth interactions.

2.1.7. Cardea

Cardea [Cardea] is a distributed authorization systems, developed as part of the NASA Information Power Grid, which dynamically evaluates authorization requests according to a set of relevant characteristics of the resource and the requester rather than considering specific local identities. Shared resources within an administrative domain are protected by local access control policies, specified with the XACML (eXtensible Access Control Markup Language) [XACML] syntax, in terms of requester and resource characteristics. Further, potential users are identified by X.509 proxy certificates, but only according to the characteristics they can reliably demonstrate. The exact information needed to complete an authorization decision is assessed and collected during the decision process itself using the Security Assertion Markup Language –SAML- [SAML]. This information is assembled appropriately and presented to the Policy Decision Point –PDP- that returns the final authorization decision for the actual access requests together with any relevant details.
Figure 10 shows Cardea's architecture. The system is currently implemented in JAVA and contains a SAML PDP, one or more Attribute Authorities, one or more Policy Enforcement Point –PEP-, one or more references to an Information Service (IAS), an XACML context handler, one or more XACML Policy Access Points –PAP- and a XACML PDP. Although all these components may be co-located on the same machine to use local communication paradigms, they may also be distributed across several machines and their functionality exposed as Web Services.
Communication between components is specified directly by the XACML and SAML standards, such as the request and response formats for obtaining information. Although both standards are transport independent, the initial implementation binds these protocols to SOAP.
To preserve message integrity the body of each SOAP message is signed using OASIS XMLSig [XMLDSig] before transmission to the intended recipient. As each message is signed only after processing is complete, the native format of the signed content is opaque to the signing process. Therefore, no dependencies between signature and content must be supported.



SAML Attributes
SAML Authorization
XACML
LDAP/Other
1 Authorization Decision Request Received
LDAP/ Info Service
Authorization Decision Statement with appropriate support information returned
3 Query AA and XACML PDP locations
9
2 Partition AA and XACML PDP search space
SAML Policy Decision Point
Generate Authorization Decision Statement and forward to appropriate PEP
7
4 Determine attributes considered by policy
5 Query appropriate attribute values
XACML Policy Decision Point
Attribute Authority
Policy Enforcement Point
6 Make authorization request
8
Report any local identity associated with the Authorization Decision Statement
CARDEA

Fig. 10. Cardea: system architecture.

Cardea attempts to provide a new resolution to portion of the overall authorization problem that existing systems do not adequately address. The first goal is to model distributed authorization in a way that separates local identity from authorization data. Reaching this goal will permit representation and evaluation of access control decisions using strictly the attributes of the entities involved, offering also a method to provide an authorization enforcement point with sufficient information to manage access control with a variety of enforcement mechanisms.
The second system goal is supporting distributed authorization while interoperating with existing security infrastructures. This facilitates the communication of authorization decisions through standard means.
Any characteristic of the subject, the requested resource, the desired action or the current environment may be considered in the authorization decision. The model adopted by Cardea represents these attributes as SAML assertions that are passed between components. Each component is free to use the assertion data in any way it needs, such as transforming it to a different native internal format. However, when communicating the data between components, all characteristics are represented in this common format, regardless of the source.
Cardea leverages the XACML model for authorization evaluation, and SAML for obtaining assertion data used during the evaluation process. Cardea assumes that the SAML PDP that accepts the initial request is responsible for providing the final authorization decision details to the PEP. The SAML PDP depends upon the content of the initial request to determine the correct XACML PDP to evaluate the request, and then the flow of communication between entities is specified by these standards.

2.2 Certificate Validation

Trust and Validation are two important concepts to build interoperable Grid services, despite the fact there is not a clear consensus about their definition. From RFC 3280 [RFC3280] the primary goal of a Path Validation process is to verify the binding between a subject distinguished name or a subject alternative name and subject public key, as represented in the end entity certificate, based on the public key of the trust anchor (i.e. a Certification Authority). This requires obtaining a sequence of certificates that support that binding. To meet this goal, the path validation process verifies, among other things, that a prospective certification path (a sequence of *n* certificates) satisfies the following conditions:

- For all *x* in *{1, ..., n-1}*, the subject of certificate *x* is the issuer of certificate *x+1*;
- certificate *1* is issued by the trust anchor;
- certificate *n* is the certificate to be validated; and
- for all *x* in *{1, ..., n}*, the certificate was valid at the time in question.

The RFC 3280 goes beyond by introducing an "extended version" of the Path Validation procedure, where trust anchors are literally *trusted* if listed into a set of Certification Authorities given also as input to the validation algorithm. Then again this process is inherently static, cumbersome and considers that all the trust anchors into this list have the same *level of trust*.
In contrast, for the present work and revisiting the concepts given in section 1.5, the term *trust* will be used from now on when referring to the *qualitative or quantitative level* in which a set of Grid services decide to interoperate among them. In consequence, *validation* will denote the mechanisms used to provide trust for the Grid.
In this section are reviewed the OCSP protocol and the CertiVeR project, two elements widely referenced during this thesis. Also for completeness is presented the SCVP protocol, despite it is still a draft proposal.

2.2.1. The Online Certificate Status Protocol (OCSP)

Specified in RFC 2560 [RFC2560] the Online Certificate Status Protocol or OCSP was proposed to be used instead of or –as in most of the cases- in conjunction with other mechanisms like CRLs to provide timely information regarding the revocation status of a digital certificate. Even though it was initially aimed for applications carrying highly sensitive and valuable information, nowadays it is being used in a wide variety of systems ranging from Web Services to the computational Grid.
When deploying a PKI, certificate validation using OCSP may be preferred over the use of CRLs for several reasons:

- OCSP can provide more timely information regarding the revocation status of a certificate.
- OCSP removes the need for clients to retrieve the (sometimes very large) CRLs themselves, leading to less network traffic and better bandwidth management.
- Using OCSP, clients do not need to parse CRLs themselves, saving client-side complexity.
- An OCSP Responder –server- may implement billing mechanisms to pass the cost of validation transactions to the seller, rather than buyer.
- CRLs may be seen as analogous to a credit card company's "bad customer list" -an unnecessarily public exposure-.
- To a degree, OCSP supports trusted chaining of OCSP Requests between Responders. This allows clients to communicate with a trusted Responder to query an alternate Responder, saving client-side complexity.

OCSP is based on a request-response scheme in which an *OCSP Client* issues a certificate status query to an *OCSP Responder* with the following data:

- Protocol version, which currently defaults to *v1* in RFC 2560.

- Service request, including the requestor name.
- Target certificate identifier, consisting of an unordered list of certificate identifiers formed with the issuer's distinguish name hash, issuer's public key hash and finally the serial number of the certificate whose status is being requested.
- Optional extensions which may be processed by the OCSP Responder, for example to demand information about the Certificate Authority quality.

The OCSP request itself may be secured if the client uses a nonce (protection against replay attacks) and digitally signs it.
Once received by the OCSP Responder, the request is verified (i.e. digital signature), processed and a definitive response message is produced. The response message is composed of the following data:

- Version of the response syntax -which also in RFC 2560 defaults to v1-.
- Name of the responder (i.e. its URL).
- Responses for each of the certificates in a request, which consist of the following data:
    - Target certificate identifier as included in the OCSP request by the client.
    - Certificate status value, which can be any of the following definitive responses: Good, Revoked (either permanently or temporarily) and Unknown.
    - Response validity interval indicating: the time when the status information produced is known to be correct (*thisUpdate*), the time when the information about the status of the certificate will be refreshed again (*nextUpdate*) and the time when the OCSP Responder signed this response (*producedAt*).
    - Optional extensions as requested by the OCSP for each certificate.
- Optional extensions as requested by the OCSP client for the whole response.
- Signature computed across hash of the response to protect the data.

In case of an error, the OCSP Responder will respond with the corresponding status code (RFC 2560 defines the *malformedRequest*, *internalError*, *tryLater*, *sigRequired* and *unauthorized* status codes).
The signing key of the OCSP Responder is a very sensitive issue in a PKI environment and in fact, depending on the certificate being used to sign the responses, we can define three different operation modes:

- Authorized OCSP Responder mode: Based in figure 11, the OCSP signing certificate (subject *B*) is issued by the same hierarchy (subject *A*) as the one that issues the certificate being requested (subject *C*) – i.e. the OCSP service is being implicitly authorized by the CA-. In this way, the user (subject *X*) does not require to trust any specific hierarchy other than the one that is being verified in order to validate the OCSP response.

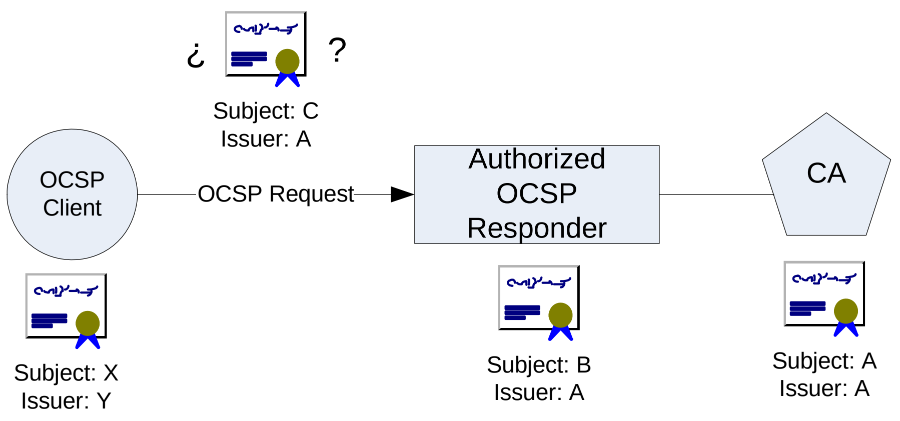


Fig. 11. OCSP Responder in Authorized mode.

- Transponder OCSP Responder mode (figure 12): some OCSP Responders are used only as gateways or proxies, forwarding the request for certificate *C* from the client (subject *X*) to a more authoritative server (which OCSP Signing Certificate's subject is any of *A*, *Y* or *W*), and then the response from this authoritative OCSP Responder back to the client. In this case request and response are not signed again. The OCSP transponder merely transports requests and responses from one side to the other.

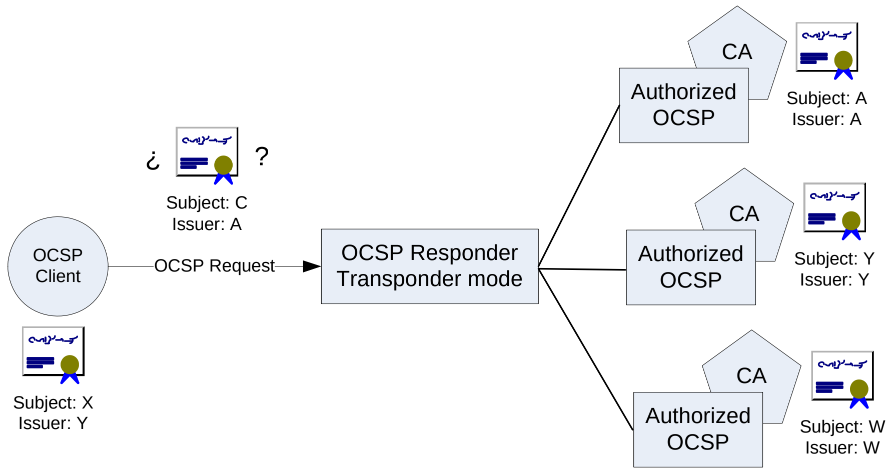


Fig. 12. OCSP Responder in Transponder mode.

- Trusted OCSP Responder mode: in this case (figure 13), the OCSP signing certificate (subject *T*) belongs to a hierarchy that differs from that of the certificates being requested (neither of *A*, *Y* or *W*) – i.e. the OCSP service has to be explicitly trusted by the user with subject *X*-.

The Trusted and Authorized modes are frequently used to centrally provide a single OCSP service connected to several different hierarchies; this feature is particularly useful for Grid's Virtual Organizations -VOs- that typically integrate users from more than one PKI hierarchy.

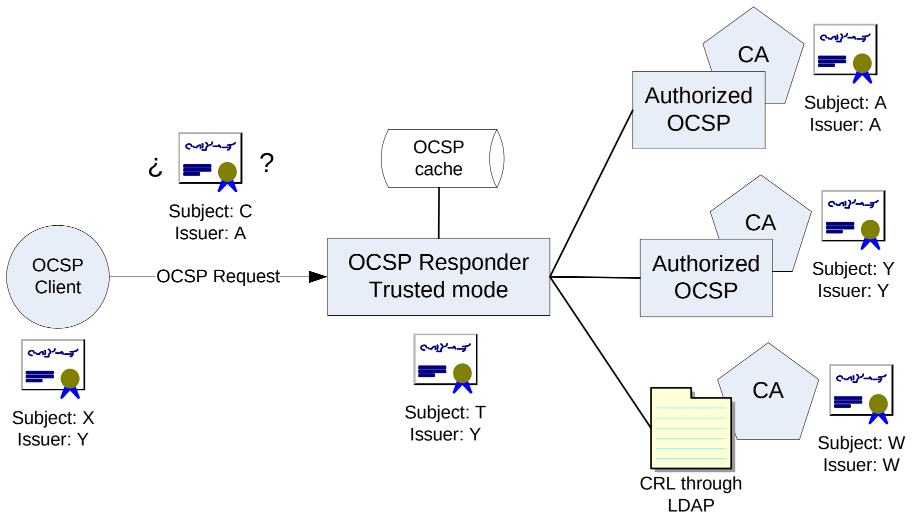


Fig. 13. OCSP Responder in Trusted mode.

In practice, OCSP Responders working under such centralized models, implement a response cache to increase their performance while reducing the number of queries to external Authorized OCSP Responders and other revocation sources (like CRLs through HTTP or LDAP).

2.2.2. Server-based Certificate Validation Protocol (SCVP)

SCVP[1] [SCVP] is a request-response protocol that allows a client to delegate certificate path construction and certificate path validation to a server. A design goal in SCVP is to leverage certificate path construction's overhead to specialized servers. In applications where minimum latency is critical, delegating validation to a trusted server can offer significant advantages. The time required to send the target certificate to the validation server, receive the response, and authenticate the response, can be considerably less than the time required for the client to perform certification path discovery and validation. Even if a certification path were readily available to the client, the processing time associated with signature verification for each certificate in the path might (especially when validating very long paths or using a limited processor) be greater than the delay associated with use of a validation server.

Two classes of applications are quite appropriate for using SCVP. In first place are those systems that want just two things: confirmation that the public key belongs to the identity named in the certificate and confirmation that the public key can be used for the intended purpose. Such clients can completely delegate certification path construction and validation to a *trusted* SCVP server. This is often referred to as delegated path validation (DPV).

There is a second group of potential users: applications that in fact can perform certificate path validation, but lack of a reliable or efficient method of constructing a valid certification path. Such clients delegate certification path construction to an *untrusted* SCVP server, but not validation of the returned certification path. This is often referred to as delegated path discovery (DPD). DPD provides several benefits: for example, a single query to a server can replace multiple repository queries, and caching by the server can reduce latency. Another benefit to the client system is that it need not incorporate a diverse set of software to interact with various forms of repositories,

[1] At the moment of writing this thesis, SCVP was still an IETF draft.

perhaps via different protocols, nor to perform the graph processing necessary to discover certification paths, separate from making the queries to acquire path validation data.
It is important to notice that in both cases (DPV and DPD), certificate validation may involve querying the following revocation sources:

1. Full CRLs (or full Authority Revocation Lists);
2. OCSP responses;
3. delta CRLs; and
4. indirect CRLs.

Therefore SCVP does not exclude the use of other validation technologies.
The path construction or validation in SCVP is performed according to a *validation policy*. In SCVP the validation policy is a RFC 3379 [RFC3379] compliant document that specifies the rules and parameters to be used by the SCVP server when validating a certificate.
A validation policy is built from three components:

1. Certification path requirements, which identify a sequence of trust anchors[2] used to start certification path processing and initial conditions for certification path validation.
2. Revocation requirements: revocation information might be obtained through CRLs, delta CRLs or OCSP responses. Certificate revocation requirements are specified in terms of checks required on the end-entity certificate and CA certificates.
3. End-entity certificate specific requirements: the validation policy might require the end-entity certificate to contain specific extensions with specific types or values (it does not matter whether they are critical or non-critical).

The policy to be used by the SCVP server can either be fully referenced in the request by the client (and thus no additional parameters are necessary) or it can be referenced in the request by the client with additional parameters.
Policy definitions can be quite long and complex, and some policies may allow for the setting of a few parameters. The request can therefore be very simple if an OBJECT IDENTIFIER (OID) is used to specify both the algorithm to be used and all the associated parameters of the validation policy. The request can be more complex if the validation policy fixes many of the parameters, but allows the client to specify some of them. When the validation policy defines every parameter necessary, an SCVP request needs only to contain the certificate to be validated, the referenced validation policy, and any run-time parameters for the request.
A server publishes the references of the validation policies it supports. When these policies have parameters that may be overridden, the server communicates the default values for these parameters as well. The client can simplify the request by omitting a parameter from a request if the default value published by the server for a given validation policy reference is acceptable. However, if there is a desire to demonstrate to someone else that a specific validation policy with all its parameters has been used, the client will need to ask the server for the inclusion of the full validation policy with all the parameters in the response.

2.2.3. Enabling multi-PKI OCSP Validation: the CertiVeR Project

CertiVeR [CertiVeR] is an EU funded project that offers a comprehensive validation service that, on top of providing validation information of a X.509 certificate in real time through the Online Certificate Status Protocol.

[2] Usually a trust anchor is defined as one public key, a CA name, and a validity time interval.

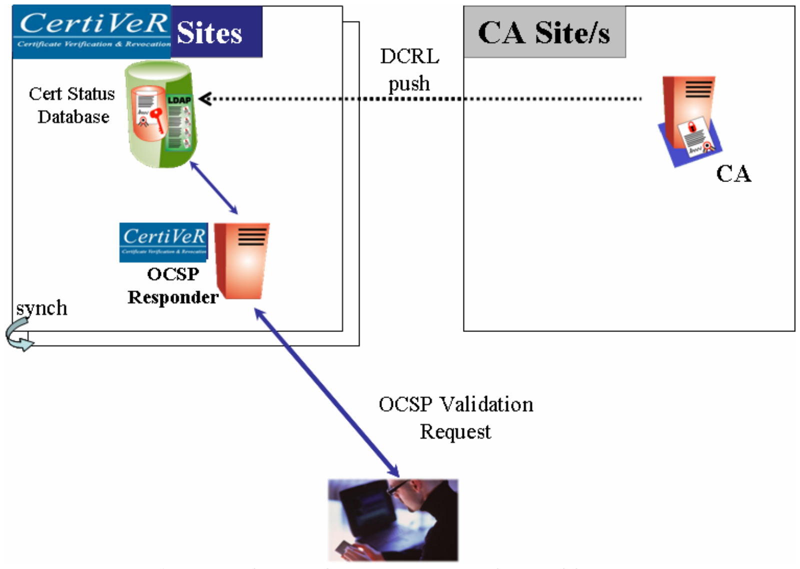


Fig. 14. The CertiVeR OCSP service architecture.

CertiVeR has the following features:

- Implementation of a CRL Updater module, which is in charge of retrieving revocation information directly from the CA's CRL through protocols like LDAP and HTTP. This information is stored in a local cache.
- Provision of a DeltaCRL connector which is used by the CRL Updater modules to remotely push any new revocation information from the remote CA into the Cert Status DB (figure 14). This connector is part of the CertiVeR service an does not require any change on the side of the CA.
- A customizable set of extensions on the OCSP response. In this way, CertiVeR can report information at several levels, such as technological – e.g. the reliability of the degree of trust in the issuing authority of the certificate (i.e. Gold: highly trusted registry and revocation procedures, Silver: highly trusted registry procedures, Bronze: low confidence registry procedures) – or commercial – e.g. information provided by the Chambers of Commerce about a company. Such type of information may dramatically increase security and e-Trust.
- CertiVeR OCSP Responder can be configured in trusted or authorized mode.
- Fault tolerance (through replication techniques, backup sites and load balancers) and high performance (using cryptographic hardware) are also provided for those organizations requiring them.

CertiVeR's multi-PKI validation features were the starting point for the "high-level OCSP Responder" infrastructure proposed in this research, just as explained in section 3.5.

2.3 Trust and Grid Services

As mentioned in [CGTrust], traditional security mechanisms typically protect resources from malicious users by restricting access to only authorized entities. However, in many situations within distributed applications one has to protect oneself from those who offer resources so that the problem is in fact reversed. For instance, a resource providing information can act deceitfully by providing false or misleading information, and

traditional security mechanisms are unable to protect against this type of threat. As noted in [Jos07], trust systems can provide protection against such threats.
Grandison and Sloman [Gra00] have defined a trust classification as a useful way of categorizing the literature related to trust in Internet services. In their work, trust is specified in terms of a relation between a *trustor*, the subject that trusts a target entity, and a *trustee*, the entity that is trusted. The following classes of trust are defined:

- **Service Provision Trust** describes the relying party's trust in a service or resource provider. The trustor trusts the trustee to provide a service that does not involve access to the trustor's resources.
  This type of trust is essential for Grids, and can be seen as a minimal trust requirement in dynamic Virtual Organizations. Many Grid applications assume this type of trust implicitly; a partner in a VO presupposes a service provision trust as a result of participating in VO, although the VO does not provide mechanisms to enforce it. In general, service provision trust is related to the *reliability* or the *integrity* of the trustee.
- **Resource Access Trust** describes trust in principals for the purpose of accessing resources owned by the relying party. A trustor trusts a trustee to use resources that he own or controls. Resource access trust has been the focus of security research for many decades, particularly on mechanisms supporting access control. Generally, resource access trust forms the basis for specifying authorization policies, which then are implemented using access control mechanisms, firewall rules, etc.
  [Gra00] highlights the distinction between trusting an entity to read or write a file on your server and trusting an entity to execute code within your workstation. Simple file access requires that the trustee will follow the correct protocol, will not divulge information read, and will write only correct data, etc. Allowing an entity to execute code on your workstation implies much higher level of trust. The code is expected not to damage the trustor's resources, to terminate within reasonable finite time and not to exceed some defined resource limits with respect to memory, processor time, local file space, etc.
- **Delegation Trust** denotes the case when a trustor trusts a trustee to make decisions on his behalf, with respect to a resource or service that the trustor owns or controls.
  Although delegation is conceptually simple, designing and deploying it within a Grid environment has proved to introduce problems regarding security. A critical point in this scenario is the level of trust assumed when delegation is employed.
- **Certification Trust** is based on the certification of the trustworthiness of the trustee by a third party, so trust would be based on a criteria relating to the set of certificates presented by the trustee to the trustor.
  Trust systems that derive certification trust are typically authentication schemes such as X.509. This class of trust is called "authentication trust" in Liberty Alliance [Lin04] and "identity trust" in [Jos07]. Grandison [Gra00] views certification trust as a special form of service provision trust, since the Certificate Authority is in fact providing a trust certification service; however Josang [Jos07] views certification trust and service provision trust as two layers on top of each other, where provision trust normally cannot exist without certification trust; in the absence of certification trust, it is only possible to have a baseline provision trust in an entity.
- **Context Trust** describes the extend to which the relying party believes that the necessary systems and institutions are in place in order to support the transaction and provide a safety net in case something should go wrong. It refers to the base context that the trustor must trust. This type of trust is called infrastructure trust in [Gra00] and context trust in [Jos07].

The main motivation behind Grandison and Sloman's classification is to define classes of high-level trust specifications, which may be further refined to low-level implementation policies, such as policies about access control, authentication and encryption. Gambetta [Gam00a] has highlighted that to make a society prosper, one needs rules (both written and unwritten), understanding of good and bad behavior with its consequences and accountabilities, initial trust and earned trust, identification of the risks associated with transactions, and so on. A similar view should be taken if we want to achieve a secure Grid society. Many of the rules of the secure Grid society can be expressed in the form of trust specifications, which can consequently be refined into policies.

For the purposes of this thesis, Certification trust will play a very important role because of its relationship with Certification Authorities, core of the Grid Security Infrastructure (GSI) and widely exploited in production Grids.

The rest of this chapter reviews three important pillars associated with trust in Grid Services: OGSA Security, WS-Security and finally the Policy Management Authorities.

2.3.1. OGSA Security

To address the specific security requirements of Grid Services (see section 1.3), the OGSA Security Group has proposed an architecture leveraging as much as possible from the Web Services Security specifications [OGSASEC]. As 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. Therefore in a Grid environment, security mechanisms should be *dynamic*.

Exposing security functionalities as services (i.e. with a WSDL 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. [OGSASEC] 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 (just like the ones mentioned in section 1.3.4), 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 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.

From the services mentioned above it is noticeable the lack of a *validation service*, able to provide assurance to interoperable Grid Services by bridging the gap between the authentication and authorization services through the concept of trust. A proposal for this kind of Validation Service will be developed in the following chapters, but first should be introduced the concept of Policy Management Authority, an administrative entity currently providing interoperability for production Grids worldwide.

2.3.2. Web Services Security

Web services offer an interoperable framework for stateless, message-based and loosely coupled interaction between software entities. These entities can be spread across different companies and organizations, can be implemented on different platforms, and can reside in different computing infrastructures. Web services expose functionality via XML messages, which are exchanged through the SOAP protocol. The interface of a Web service is described in detail in an XML document using the "Web Service Description Language" (WSDL).

In order to provide security, reliability, transaction abilities and other features, additional specifications exist on top of the XML/SOAP stack. The creation of the specifications is a cross-industry effort, with the participation of standardization bodies such as W3C and OASIS. A key element in the Web services specifications is the so-called combinability. Web services specifications are being created in such a way that they are mostly independent of each other, however they can be combined to achieve more powerful and complex solutions. Based on [Geu05] the following paragraphs describe some individual specifications, with a special focus on those dealing with trust and security.

The *WS-ReliableMessaging* specification describes a protocol for reliable delivery of SOAP messages in the presence of system or network failures. To do so, the initial sender retrieves a unique sequence identifier from the ultimate receiver of the sequence to be sent. Each message in the sequence is uniquely bound to that identifier, together with a sequence number. The receiver of the sequence acknowledges the sender what messages have already been received, thus enabling the sender to determine based on the sequence number which messages have to be retransmitted. WS-ReliableMessaging should be used in conjunction with WS-Security, WS-Secure-Conversation and WS-Trust in order to provide security against attackers at the network layer.

The Web Services Policy Framework, *WS-Policy*, provides a general-purpose model to describe web service related policies. WS-Policy by itself only provides a framework to describe logical relationships between policy assertions, without specifying any assertion. *WS-PolicyAttachment* attaches policies to different subjects. A policy can be attached to an XML element by embedding the policy itself or a link to the policy inside the element or by linking from the policy to the subject that is described by the policy. WS-PolicyAttachment also defines how policies can be referenced from WSDL documents and how policies can be attached to UDDI entities and stored inside a UDDI repository. *WS-MetadataExchange* defines protocols to retrieve metadata associated with a particular web services endpoint. For example, a WS-Policy document can be retrieved from a SOAP node using *WS-Metadata*. *WS-PolicyAssertions* specifies some common WS-Policy assertions, related to text encoding, required SOAP protocol version and so-called “MessagePredicate” assertions that can be used to enforce that a particular header combination exists in a given SOAP message.

WS-SecurityPolicy defines certain security-related assertions that fit into the *WS-Policy* framework. These assertions are used by *WS-Security, WS-Trust* and *WS-SecureConversation*. Integrity and confidentiality assertions identify the message parts that have to be protected and it defines what algorithms are permitted. For instance, the “SecurityToken” assertion tells a requestor what security tokens are required to call a given Web service. Visibility assertions identify what particular message parts have to remain unencrypted in order to let SOAP nodes along the message path being able to operate on these parts. The “MessageAge” assertion enables entities to constrain after what time a message is to be treated as expired.

The WS-Security specification defines mechanisms for integrity and confidentiality protection, and data origin authentication for SOAP messages and selected parts thereof. The cryptographic mechanisms are utilized by describing how XML Signature and XML Encryption are applied to parts of a SOAP message. That includes processing rules so that a SOAP node (intermediaries and ultimate receivers) can determine the order in which parts of the message have to be validated or decrypted. These cryptographic properties are described using a specific header field, the <*wsse:Security*> header. This header provides a mechanism for attaching security-related information to a SOAP message, whereas multiple <*wsse:Security*> header may exist inside a single message. Each of these headers is intended for consumption by a different SOAP intermediary. This property enables intermediaries to encrypt or decrypt specific parts of a message before forwarding it or enforces that certain parts of the message must be validated before the message is processed further.

Besides the cryptographic processing rules for handling a message, WS-Security defines a generic mechanism for associating security tokens with the message. Tokens generally are either identification or cryptographic material or it may be expressions of capabilities (e.g. signed authorization statements). WS-Security 1.0 does only define a simple user name token, a container for arbitrary binary tokens (base64 encoded) and a container for XML-formatted tokens. Additional specifications define various 'token profiles' that introduce special token formats. For instance, the 'WS-Security X.509 Certificate Token profile' defines how X.509 certificates, certificate chains or PKCS#7 certificate revocation lists may be used in conjunction with WS-Security.

The *WS-Trust* specification introduces the concept of "security token services" (STS). A security token service is a Web service that can issue and validate security tokens. For instance, a Kerberos ticket granting server would be an STS in the non-XML world. A security token service offers functionality to issue new security tokens, to re-new existing tokens that are expiring and to check the validity of existing tokens. Additionally, a security token service can convert one security token into a different security token, thus brokering trust between two trust domains. WS-Trust defines protocols including challenge-and-response protocols to obtain the requested security tokens, thus enabling the mitigation of man-in-the-middle and message replay attacks. The WS-Trust specification also permits that a requestor may need a security token to implement some delegation of rights to a third party. For instance, a requestor could request an authorization token for a colleague that may be valid for a given time interval.

WS-Trust uses WS-Security for signing and encrypting parts of SOAP messages as well as WS-Policy/SecurityPolicy to express and determine what particular security tokens may be consumed by a given Web service. WS-Trust is a basic building block that can be used to rebuild many of the already existing security protocols and make them fit directly in the web services world by using Web service protocols and data structures.

WS-Federation introduces mechanisms to manage and broker trust relationships in a heterogeneous and federated environment. This includes support for federated identities, attributes and pseudonyms. "Federation" refers to the concept that two or more security domains agree to interact with each other, specifically letting users of the other security domain accessing services in the own security domain. For instance, two companies that have a collaboration agreement may decide that employees from the other company may invoke specific web services. These scenarios with access across security boundaries are called "federated environments" or "federations". Each security domain has its own security token service(s), and each service inside these domains may have individual security policies. WS-Federation uses the WS-Security, WS-SecurityPolicy and WS-Trust specifications to specify scenarios to allow requesters from the one domain to obtain security tokens in the other domain, thus subsequently getting access to the services in the other domain.

Some of the requirements presented in the analysis of requirement through the VO lifecycle can be met by application of Web services specification, as shown in [Are05].

The identification phase includes defining VO wide policies as well as selecting potential business partners who are both capable of providing the required services and of fulfilling the trustworthiness requirements of the VO. The selection of potential business partners involves looking at repositories, which can realize. The usual Web

service technology to be applied is *WSDL/UDDI*, WSDL describes messages and operations while UDDI offers a discovery mechanism. To include the provision of SLA, “Web Service Level Agreements” (*WSLA*) has been developed, a XML language for specifying and monitoring SLA for Web Services, which is complementary to WSDL. Determining the required service providers and a proper negotiation requires secure communication. The WS-Security specification and data origin authentication for SOAP messages can be used between the entities to secure the communication.
The realization of the VO requires the creation of federations, where two or more security domains agree to interact with each other, specifically letting users of the other security domain accessing services in the own security domain. The WS-Federation specification deals with federations by providing mechanism to manage and broker trust relationships in a heterogeneous and federated environment. This includes making use of WS-Trust to support for federated identities, attributes and pseudonyms. The dissemination of configuration information requires secure communication as provided by the WS-Security specification.
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 –i.e. an unauthorized access detected by the access control systems- and security threats –i.e. an event detected by an intrusion detection system- need to be notified to other members in order to take appropriate actions. VO members will also need to enforce security at their local site. For example, providing access to services and adapting to changes and the violations. Monitoring can be supported by event management and notification mechanisms using the WS-Eventing and WS-Notification specifications. This allows the monitoring service partner to receive messages when events occur in other partners. A mechanism for registering interest is needed because the set of Web services interested in receiving such messages is often unknown in advance or will change over time.

2.3.3. Grid interoperability through Policy Management Authorities

As Grid computing became more popular also Virtual Organizations proliferated at the same rate, and this finally resulted in the breed of several Certification Authorities (a common practice as each organization installing a Grid environment was also used to define its own Certificate Authority). Soon this represented a big interoperability problem between the users and resources belonging to different institutions: their computing resources where in different domains, but the need of cooperation through a Grid environment required to share them all.
A suitable solution for this problem was given by the Policy Management Authorities (PMAs) which define a minimum set of security provisions –in the form of an Authentication Profile as in [GridAP]- that must be accomplished by all the Grid-PKIs wishing to interoperate between them. The PMA’s Authentication Profile document describes the minimum set of requirements posed on Certification Authorities (CA) that traditionally issue long-term X.509 credentials to end-entities, who will themselves posses and control their key pair and their activation data. These CAs act as an independent trusted third party for both subscribers and relying parties within the infrastructure.
In general, any CA requesting membership to a PMA must present its Certification Policy (CP) document which will be in turn carefully evaluated by experts against a set of rules or minimum security provisions defined into the PMA’s “Authentication Profile” thus determining whether if it is fulfilled or not. This is performed just once when the new Grid-PKI wishes to be part of an existing PMA and it is commonly called

the *"accreditation process"*. Obviously a further auditing process could determine if in practice the CP itself is also being satisfied by the PKI, but in most of the cases this audit is out of the PMA's scope.
In summary a PMA:

- Defines and issues minimum requirements and best practice documents; these minimum requirements (also known as Authentication Profile) may govern any aspect of certificate issuance and reliance.
- Maintains and revises these documents according to current developments.
- Accredits Certification Authorities with respect to the authentication profile.
- Accredits Certification Authorities only for those Grid applications that relate to inter-organizational distributed resource sharing in a scientific context.
- Is primarily concerned with Grid communities in their region.
- Harmonizes and synchronizes minimum requirements with other regional PMAs.
- Fosters trust relations for authentication purposes within the context of Grid communities.

Three regional Grid PMAs currently exist around the globe (Europe's EUGridPMA [EUGridPMA], America's TAGPMA [TAGPMA] and Asia-Pacific's APGridPMA [APGridPMA]) and nowadays they are being harmonized and synchronized through the International Grid Trust Federation –IGTF [IGTF]-.

2.4 Open issues

Despite the security advantages brought by current authentication and authorization technologies to the computational Grid, there are still open issues related with trust management and user's validation that must be solved in production environments requiring high levels of assurance, for example eHealth systems.
In an effort to summarize the main Grid-related authentication and authorization features of the technologies reviewed in this section, table 1 presents a comparison of them.
From table 1 it is easy to notice in the reviewed technologies the lack of features related with *trust* and *validation* (Levels of Authentication, support to dynamic VO and real-time validation), which are central for building interoperability relationships in Grid environments.
On the validation arena we have found a lack of systems implementing real-time checking of end user's credential status, due mainly because of the complex deployment issues that arise when trying to adapt existing mechanisms and protocols to the computational Grid. This is one of the reasons to continue using "traditional" schemes like Certificate Revocation Lists (CRLs) in current Grid installations around the globe.
In this chapter we have also overviewed the importance of the trust management concept in Grid environments, where interoperability is a *must* for creating Virtual Organizations grouping users and resources from multiple institutions. Despite the efforts towards achieving this goal (i.e. with the Policy Management Authorities), current procedures and mechanisms are poorly automated (let us mention as an example the PMA's accreditation process), therefore providing little or none information at all to other processes about security features related with the user's authentication and identity assessment process, that is, its *level of assurance*.

Table 1. AAI technologies: a comparison for building Grid interoperability

	Architecture (1)	**PKI Required?** (2)	**Levels of AuthN** (3)	**Dynamic VO** (4)	**RT Validation** (5)	**Grid Integration** (6)
GSI	D	Y	N	N	N	Y
VOMS	D	Y	N	N	N	Y
CAS	C	Y	N	N	N	Y
PRIMA	C	Y	N	N	N	Y
Akenti	D	Y	N	Y	N	Y
PERMIS	D	Y	N	N	N	Y
Shibboleth	D	Y	N	N	N	Y
Cardea	D	Y	N	N	N	Y

(1) Denotes if its components are Centralized (C) or Distributed (D) along the Grid.
(2) Requires the use PKI technology (Y)es or (N)o.
(3) Different mechanisms for entity's authentication (therefore different levels of assurance), (Y)es or (N)o.
(4) Allows building dynamic, "on-the-fly" Virtual Organizations, (Y)es or (N)o.
(5) Entity's authentication credentials are validated in real-time, (Y)es or (N)o.
(6) May be seamlessly integrated into Grid installations, (Y)es or (N)o.

A common belief into the Grid community is that this data becomes quite important nowadays to take authorization decisions dependent on the security level associated with the requestor identity. This argument has been already demonstrated by surveys like [LoA].

To cope with the problems mentioned above and then "bridge" the gap between the Grid authentication and authorization processes, the rest of this thesis presents our contributions to build a Grid-OCSP protocol and moreover, to adopt a methodology able to quantitatively evaluate a user's level of security, thus providing the Extended Path Validation features required by these environments.

CHAPTER 3 GRID-OCSP: A CONTRIBUTION TO CERTIFICATE VALIDATION

3.1 Motivation

Grids use X.509 [X509] certificates for authentication and as a basis for authorization. A reliable, secure Grid infrastructure depends on the integrity and validity of these certificates. While X.509 certificates have built-in lifetimes, this mechanism is not adequate to deal with every aspect of certificate life cycle and management. Certificates can be lost by their owners, can be "compromised", or the justification for holding the certificate may no longer apply. In such cases these certificates need to be revoked before its expiration date is reached. Distribution of certificate revocation lists (CRLs) partly supports this need, however CRL distribution and maintenance in the Grid has proven difficult for practical reasons. In lieu of or as a supplement to checking against a periodic CRL, it may be necessary to obtain timely information regarding the revocation status of a certificate. Examples include high-value funds transfer or large stock trades.

The Online Certificate Status Protocol (OCSP) enables applications to determine the (revocation) state of an identified certificate. OCSP may be used to satisfy some of the operational requirements of providing more timely revocation information than is possible with CRLs and may also be used to obtain additional status information. OCSP provides a simple query protocol for relying parties to perform revocation check lookups on certificates, without the need to maintain up-to-date sets of CRLs from a number of different certification authorities, at each and every computer that is part of the Grid.

The evolution of security concerns in Grids provides a number of scenarios for OCSP usage. In the trust arena, the increasingly large number of Grid CAs, and the attendant need for every end point in every Grid PKI to manage a relationship with each CA in order to obtain revocation information, presents a scaling problem for designers and an implementation problem for security and system administrators.

Next are given the contributions of this thesis related with a research to deploy an OCSP architecture for the Computational Grid (a "Grid-OCSP" system), which may be able to cope with the specific requirements and challenges identified for this environment. At the time of writing this work, most of the proposed ideas were being considered as an Open Grid Forum's Recommendation [OCSPReq].

3.2 OCSP Relying Party requirements for Grids

For OCSP to be used in multi-organizational Grid environments, support for it must be integrated in any path validation software used according to the following recommendations.

3.2.1. Revocation Source Requirements

In some scenarios, CRLs provide a better mean of processing revocation than OCSP; for instance, in offline environments, or in the case of a relying party validating many different client certificates, it is better from a performance point of view to have all revocation information cached locally. In other scenarios, CRLs may be used as a backup source of revocation information in case the OCSP service fails (i.e. due to temporary network outage). As OCSP is deployed, and becomes a production service in Grids, not all CAs will be capable of providing Authorized responder service immediately. According to the preset research, relying parties and security managers will see a gradual transition from CRL to OCSP as the primary source of revocation information.

In light of the issues listed above, relying parties must be capable of handling both CRLs and OCSP, and it must be a configurable option which source of revocation to prefer and which to use as a backup, on a per-issuer basis.

CRLs revocation reason codes and their usage are largely outside the scope of this thesis. However, one code needs to be discussed: *certificateHold*, which was provided in the [X509] and [RFC3280] with little guidance in how it was supposed to be used. Grid applicability has been suggested in a couple of use cases where temporary blacklisting is needed. The research done for this thesis suggest, on an experimental basis, the use of *certificateHold* to suspend a given certificate for a time; the manager of the CRL may later decide to remove the *certificateHold* status, or change the revocation status to another recognized revoked status. This constitutes a Grid-only usage. The policies that govern how *certificateHold* status is applied or how its status is changed are outside of the scope of this document.
Of all the reason codes, *certificateHold* appears to have the unique property of impermanence. It can be applied to a certificate, and then withdrawn, unlike the other reason codes (i.e. *keyCompromise*), for which no such behavior could be considered reasonable. As a result, on removal of *certificateHold* status, the prior CRLs become invalid. While CRL policy is outside the scope of this document, it seems desirable to limit the *certificateHold* reason code to OCSP usage, or limit its appearance to Delta CRLs if possible (see the Delta CRL usage subsection below). This may not be possible in a scenario that requires blacklisting a certificate for use with a legacy service unable to use OCSP, but such scenarios should take into account the possibility of stale CRLs.

3.2.2. OCSP Latency and Cautionary Period

Latency is inherent in network transactions, and revocation has a built in quality of uncertainty: Given an answer of Good, the relying party knows the certificate was Good the last time somebody created a CRL. But will the certificate still be good the next time the CRL is updated?
For trust evaluation purposes, the responder may want to quantify an upper bound and/or approximation of the current latency, and convey that to the relying party. To date, there is no well-defined technical means of augmenting an OCSP response with such "cautionary period" information.
The "traditional OCSP" currently can provide three timestamps characterizing the revocation response just ad mentioned in section 2.2.1. These timestamps are: *thisUpdate* (the time at which the status being indicated is known to be correct), *nextUpdate* (the time at or before which, newer information will be available about the status of the certificate) and *producedAt* (the time at which the OCSP Responder signed this response). If *nextUpdate* is not set, the responder is indicating that newer revocation information is available all the time. Whenever this field is set, it is an indicator that the revocation information has been retrieved from a CRL being released only at certain intervals of time. In certain cases, this interval may be quite large -even in the order of days-. Both *thisUpdate* and *nextUpdate* are from the underlying CRL timestamps.
Taking into account these parameters, we formulate the concept of *"cautionary period"* as follows:

Definition 1. We denote as "cautionary period" the interval of time during which relying parties assume a degree of uncertainly about the Proxy Certificate's status being reported by the OCSP Responder.

Given the fact that OCSP servers producing pre-computed responses (which fix a *producedAt* value for that response very close to *thisUpdate* and with not relation to the time at which the Request is being done by the OCSP client) have the potential to confuse relying parties when evaluating the cautionary period, a new timestamp called *receivedAt* is introduced to specify the moment when relying parties are receiving the OCSP Response. In terms of the mechanisms introduced in this document, *receivedAt* is given by:

- For prevalidation *receivedAtWSRF*, which denotes the time at which the Grid client's (service invoker) Proxy Certificate is being validated.
- For the second-level caching mechanism *cacheHit*, denoting the time at which the cached OCSP Response is being found and retrieved.

In this way, the cautionary period is given by the following interval:

[thisUpdate , receivedAt]

The longer such cautionary period is, the less precision resulted in the OCSP response being provided. This factor is a measure of the risk level that the relying party is willing to take when accepting the result status coming from an OCSP Response as is. Relying party's validation middleware should discard OCSP Responses conveying a cautionary period greater than the one specified in the validation policy.

3.2.3. Caching OCSP Responses

A relying party might prefer to query a local, fast responding OCSP Responder; this server manages a database of cached responses from previous queries based on lifetimes supplied in those queries, perhaps with a pre-set time-to-live (i.e. computed from a cautionary period). The expectation is that this responder is acting in Trusted mode (see OCSP review in section 2.2.1) and must be configured with a default service locator (high-level responder). This responder may poll CAs for CRL information, but this is not recommended. OCSP Responders that cache responses may pre-sign those responses for future use.

On the other hand, it is often the case that a relying party interacts with some OCSP Responder on a frequent basis. Instead of each interaction triggering a new OCSP request, causing unnecessary load and system latency, caching of responses should be allowed and may be supported by the relying party software. Enabling local caches at relying parties is recommended to reduce the number of queries made to the OCSP infrastructure, thus optimizing available resources (also consider that some OCSP providers sell their service on a per-request basis). In general should be noted that while OCSP responses reporting a permanently revoked certificate status can be cached indefinitely, it must be possible to configure the maximum size and lifetime/freshness of the entries in an OCSP response cache (for a relying party cache it is advisable to use the Cautionary Period parameter as defined in 3.2.2); this is especially important when dealing with suspended certificates -*certificateHold*-.

Further details about caching and other OCSP optimizations schemes based on our Grid-OCSP prototype are given in section 6.2.2.

3.2.4. Responder Discovery

At a minimum, the Grid-OCSP must be able to dynamically locate the OCSP Responder. Local configuration must have precedence over any service locator information located in the certificate's Authority Information Access -*AIA*- extension. OCSP relying parties should include any information about an Authorized responder location (as indicated by OCSP service locator URLs in the *AIA* extension of the subscriber certificate) in its OCSP request.

To allow for high-availability and load balancing, it should be possible to associate each issuer name with more than one responder URL. However a default responder for “all other” issuers should be configurable as well.
Any OCSP server acting as Clearinghouse or Trusted Responder must be contacted first. When the OCSP relying party is provided with more than one Responder URL or even with more than one revocation source, then it must follow the recommendations given in section 3.2.6 to obtain a definitive status code.

3.2.5. Nonce

A nonce is used to bind a specific query to a specific response, preventing replay attacks (see [RFC2560], section 4.4.1). In practice it has been used to force an OCSP Responder to issue a fresh response. Large-scale OCSP Responder implementations continuously pre-produce OCSP responses in the background, for maximum throughput.
Pre-produced OCSP responses cannot be used to service nonced requests, because they require a signed answer from the responder, including the transaction-unique nonce. Therefore, for efficiency reasons, OCSP relying parties may decide not to make use of nonce in the requests. This decision should balance efficiency, availability of dynamic information and replay attacks prevention.
Forcing a fresh response at first glance seems like a good way for a relying party to insure an up-to-date, real-time validation check. However, OCSP does not specify responder behavior in sufficient detail to guarantee this behavior; the nonce requires it to do additional work, including an expensive digital signing that adds to the latency of the response. Nonce should not be used in Grid configurations, except where circumstances and replay vulnerability are a particular concern.

3.2.6. Error Handling and the Unknown Status Code

OCSP relying parties will occasionally encounter errors from their network stack, errors returned from the OCSP Responder, or the *Unknown* status code: in most cases, such a failure means “try somewhere else” or “try later”, indicating to the relying party to probe other possible sources (including other OCSP Responders) for revocation information, possibly a bit later in time. If, after an exhaustive search of all available sources of revocation information, the status of a certificate is still *Unknown,* the relying party will have to choose between three unpalatable alternatives: interpret “network error or Unknown” as either *Good* or *Revoked*, or even trigger a fail-back mechanism (i.e. allowing to wait a predefined timeout and start the whole search process over again). Interpretation of an “Unknown” as a definitive *Good* leads to worry and conflict with security policy, so a final *Revoked* status is the choice of most security experts should all other alternatives fail. In practice, this policy may lead to an unnecessary denial-of-service so it is recommended to implement high availability mechanisms at both the OCSP Server and the relying party, just as the ones proposed in section 6.2.2.
In light of these issues, relying parties must be provided with a configuration option to allow local rules and security policies to be applied in circumstances where the revocation status is undetermined.
The relying party should perform additional checks before accepting an OCSP response; let us take for example verifying the nonce returned against the nonce sent (if a nonce was used), validating the OCSP Response’s signature (noticing the use of the *ocsp-nocheck extension*) and verifying its freshness (checking that the *producedAt* timestamp is within the allowed lifetime or cautionary period – this is especially important in the case of cached or preproduced responses).

3.2.7. OCSP Request Signing

The relying party may choose to sign the OCSP Request, in which case performance and security implications must be considered. In general, to allow a wide range of customizable options this research recommends using an OCSP configuration file. In Chapter 6 will be introduced OGRO, a prototype developed with this feature.

3.3 OCSP Responder Requirements

In this section are explained the requirements for a Grid-OCSP Responder, mostly based on the results of the research performed with the CertiVeR's validation service (explained in section 2.2.3).

3.3.1. Performance, Key Protection

Most OCSP Responder implementations can use cryptographic hardware to further accelerate their performance and also to provide additional protection of the signing keys. The OCSP signing key is the second-most important key in a PKI, just after the CA's even though some experts consider it to be the most important, since it is used to assert the current status of an issued certificate. Moreover, by its very nature, OCSP Responders are exposed to network connectivity, while most CAs work off-line.

While our research does not enforce the use of hardware protection, it does recommend that the security of the OCSP Responder key should be in parity with the CA issuing key.

3.3.2. Responder Certificate

An OCSP Responder implementation must support live updates of its signature key material. It is important to consider that due to the sensitivity of relying parties to network errors, service outages and disruptions must be avoided while updating the OCSP Responder's signing keys.

Authorized Responder scenarios (see section 2.2.1) may be implemented by enabling the Responder to use (with the same key pair) multiple signing certificates simultaneously: one for each issuer for which the OCSP Responder provides revocation information.

3.3.3. Revocation Information Sources

An OCSP Responder must consider several sources and formats of revocation information. These include CRLs, databases, and simple files (blacklisted users or Grid services); it may also include several retrieving protocols (i.e. LDAP, HTTP, HTTPS, FTP, etc.). The sources must be secure and trustable. Disclosure of access controls and roles for such services is expected as part of the policy disclosure for OCSP services. It is also expected that revocation or blacklisting events in such services will be authenticated, authorized, and logged with the same care and detail provided by CRL update operations in Certification Authorities.

Non-CRL databases such -as blacklists- must provide revocation information in accordance with [RFC2560] expectations and requirements. In particular, these databases must include a *thisUpdate* attribute value assertion; should include a *nextUpdate* assertion and should provide the equivalent to CRL Entry Extensions (also known as "CRL reason codes") as defined in [RFC3280]. This is necessary in order to provide a uniform presentation to the relying party.

In general, errors from external revocation services must be avoided, and when this is not possible, they must be carefully logged, and in some cases they may need to trigger alarms. A suppressed error from a failed local blacklist service might inadvertently allow a security breach.

3.3.4. Transponder Mode

While not part of the OCSP standard, it is common in real-world deployments for an OCSP Responder to act in Transponder mode, in other words it will act as a proxy/gateway and forward any relying party requests as-is to other, more authoritative, Responders, and likewise transparently forward the response back without re-signing it. Transponder mode is useful for organizations that want to control the outbound OCSP traffic originating from within their own administrative domain(s). When combined with an OCSP response cache, a local responder in Transponder mode can be useful to cut response latencies, reduce network traffic, and minimize load on external Responders.

In addition, a Responder in Transponder mode can be used for load balancing, to route the OCSP request according to specified parameters (i.e. OCSP load, network traffic, availability, etc.), or as a central "redirector", thus consolidating configuration and management issues.

3.4 Certificate Authority Requirements

3.4.1. Certifying Authorized Responders

A CA's Authorized Responder should be issued a certificate with the *ocsp-nocheck* extension as well as *OCSPSigning* in the *extendedKeyUsage* extension, as described in [RFC2560]. Direct use of the CAs private key for OCSP Response signing is not recommended and must be avoided. As a result, the Authorized Responder certificate is not subject to revocation check through OCSP. In order to reduce the risk associated with the Responder certificate, it is suggested that the Authorized Responder certificate have a very short lifetime.

Current CA practice in Grid-PKIs suggests that Authorized Responders will be difficult for many CA providers to operate. Many Grid-CAs are offline services, running on an as-needed basis (perhaps once a day to produce a CRL). Very short lived certificates, with lifetimes in minutes, and an always-available OCSP server, may be quite difficult for some CA providers to deploy. In the case of an infrequently-operated offline CA, it is recommended for the CA to produce a large number of signed OCSP certificates, whose key pairs are securely managed, in order to cover the time gaps between its operations. This is an ideal scenario for secure key storage devices.

CA providers who cannot easily operate an OCSP Responder service with practically zero downtime can sign an OCSP Responder certificate for another OCSP Responder, with the desirable reliability characteristics.

3.4.2. Certificate Revocation Lists

It is recommended that CRLs be type CRLv2, in order that revocation reason codes can be made available for use in OCSP responses. If a CA provides or supports OCSP services (such as providing an OCSP locator reference in end-entity certificates), the CA must continue issuing at least a full CRL at regular intervals, in order to provide relying parties a reasonable fallback strategy.

Revocation data from CRL and non-CRL information sources must be made available to the OCSP Responder in a timely and reliable way.

3.4.3. Delta CRLs
CAs should provide Delta CRLs (see [RFC3280] section 5, for a full discussion of CRL issues). Delta CRLs provide two key benefits that can greatly enhance the quality of an OCSP service:

- The smaller size of a delta CRL reduces the network load, and the amount of processing time required by the OCSP Responder to integrate the new revocation information into an existing database.
- Delta CRLs can also greatly enhance the timeliness of revocation information based on CRLs. Delta CRLs come with their own *nextUpdate* field. OCSP Responders capable of managing Delta CRLs will be able to assemble a new effective CRL, and report a new effective *nextUpdate* from the information provided by the Delta CRL.

Both of these features, while relatively unimportant in current Grid-PKIs, will improve the responsiveness and minimize latency in Grid deployed OCSP Responders.
In Grid PKIs today, CRLs are issued with fairly large *nextUpdate* periods (between 15 and 30 days are not uncommon). A *nextUpdate* field set far in the future allows sufficient time for most temporary network or host problems to be resolved before Grid end users are affected. This resolution of the problem does not meet current security requirements very well (see also the recommendation for defining a cautionary period in section 3.2.2)
Careful production of Delta CRLs and base CRLs will provide better quality, lower latency revocation information to OCSP Responders (in particular to Trusted ones and sophisticated relying party's implementations), while shielding legacy CRL users and lightweight implementations from the additional complexity.

3.4.4. CRL Latency
In order to minimize latency effects and provide the most current information available to relying parties, our research found that CAs should push (publish) CRL information immediately to such entities as soon as a revocation takes place. Obviously CAs will not know about the existence of every Responder in the world, so at least Authorized Responders for that particular CA should be updated in this fashion.
It is also advisable for CAs to develop a relationship with one or more "clearinghouses" or large VO Trusted Responders. Section 3.5.1.2 provides further details about this particular configuration.

3.5 OCSP Service Architecture
OCSP Responder deployments suitable for Grids can be viewed from two different points of view: the relying party expectations and the OCSP Responder viewpoint. Let us consider in this section both aspects and how they might be interpreted according to the OCSP Responder capabilities.

3.5.1. Relying party expectations
3.5.1.1.Site or Organizational Responders
Local Responders, or even relying parties, may contact a site or Virtual Organization Responder. This service will contain a full range of policy rules, path validation rules, and certificate validation rules appropriate to the site or organization and may be configured with one or more default service locators.
This OCSP Responder also should be acting in Trusted mode, but may act in Transponder mode if necessary. VO and Site responders should poll and retrieve CRL information from CAs, whereas this data is not directly published.

3.5.1.2.OCSP Clearinghouse

Based on current operations, it is recommended that the emerging regional Grid PMAs and even the IGTF [IGTF] itself consider creating several well-known OCSP “clearinghouse” Responders. Middleware discovery software should look for these well-known responders when no other methods succeed. Since the operational and financial issues of such services are considerable, the details must be determined elsewhere; but without such services the viability of revocation services in Grids is in doubt. These clearinghouse Responders will operate in Trusted mode and it is expected for them to gather CRL information from every known CA used by Grid PKIs. In order for such a service to be trustable by Grid relying parties, signing certificates may have to be publicly and securely distributed in a fashion similar to that currently provided by Grid CAs.

By concept, OCSP Responders must provide the best quality revocation information to their relying parties. Responders above the level of a local caching service will have access to a variety of sources of revocation information about each certificate. Responders acting as intermediaries must adjust their search strategies on a case-by-case basis to provide the best quality information.

3.5.2. OCSP Responder viewpoint

An overall OCSP service architecture suitable for Grids is depicted in figure 15. It consists of four classes of components:

- Authorized OCSP Responders.
- Trusted OCSP Responders.
- OCSP Transponders.
- OCSP relying parties.

All of the components from figure 15 may appear in multiple instances and should be configurable according to the needs of their operators. A typical site (organization or administrative domain) configuration includes at least two (for fault tolerance) Trusted Responders, configured to serve certificate status information for every CA accepted by the site, either by downloading the appropriate CRLs or by sending OCSP Requests to an external Responder whenever such exists (indicated by local configuration or presence of a service locator URL in the incoming OCSP Request, in turn copied by the OCSP relying party from the subscriber certificate that is currently being validated). In the latter case, the Trusted Responders may act in Transponder mode and CRL information may be used as a fallback source of revocation information (as indicated by path *#1* in figure 15).

As indicated by path *#2* in the figure 15, it is foreseen that “related CAs”, for instance subordinate CAs or those members of the same PMA, may join efforts and set up a shared Trusted Responder serving requests on behalf of all of the CAs (note that such Responder can be Authorized only if every member of the PMA provides it with an OCSP signing certificate).

To alleviate the need for complex configurations at each site, or allow for load balancing, an OCSP Transponder may act as a redirector for some CAs as indicated by path *#3* in figure 15. Note that in this case the OCSP Transponder is not re-signing the responses.

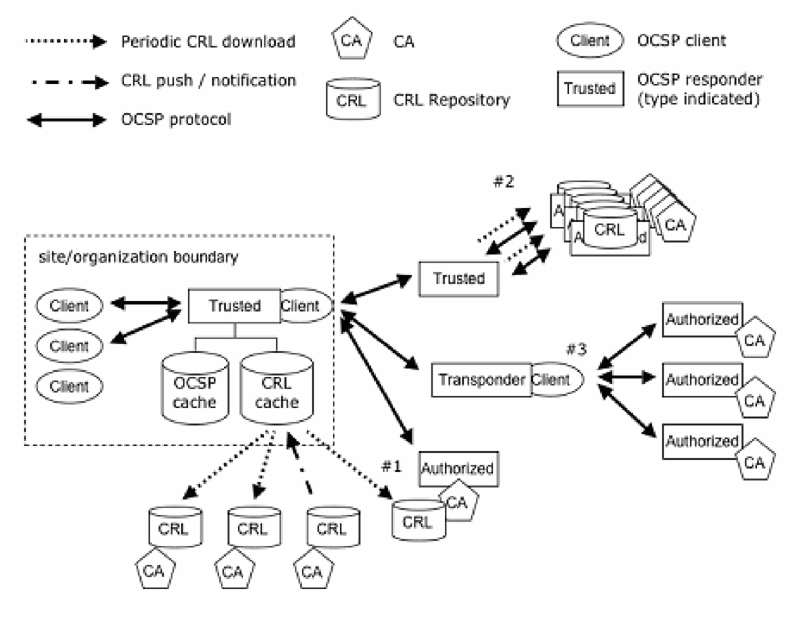


Fig. 15. An overall Grid-OCSP Responder architecture.

3.5.2.1.Authorized Responders

The Authorized Responder is the most typical OCSP configuration, arguably the most scalable and maintainable. The CA delegating this authority is directly responsible for providing timely updates to its Authorized Responders.

However Authorized Responders create a circular dependency problem in that relying parties may wish to know whether the responder certificate itself has been revoked or not, before accepting the signature of the OCSP Response. To address this and other similar problems, a certificate can be marked with the *ocsp-nocheck* extension, indicating to relying parties not to attempt to verify the revocation status of it (at least not using OCSP).

The *ocsp-nocheck* extension is typically combined with relatively short-lived certificates and close supervision of the issuer. The use of hardware key protection can mitigate some, but not all of the concerns with the Responder key-pair, since the fundamental question of its revocation is being side-stepped. The *ocsp-nocheck* extension can also be used to mitigate anticipated load problems: for instance, the TLS certificate of a popular (but low-risk) server may be stamped with the *ocsp-nocheck* extension to eliminate OCSP queries otherwise triggered by all relying parties connecting to that server.

The operator of an Authorized Responder should ensure high availability of the OCSP service, for instance by operating several independent instances of the Responder or making use of fault-tolerant or mirrored systems.

3.5.2.2.Trusted Responders and Transponders

The Trusted Responder is one that may be particularly useful in scenarios requiring additional controls and capabilities. Trusted Responders are usually operated by sites or at an organizational level to provide service to local OCSP clients, thus centralizing the complexity of the OCSP/CRL configuration and minimizing the need for providing means to control outgoing network connectivity.

It is feasible to think that combinations of Authorized and Trusted Responders are likely to be deployed in Grids. Trusted Responders should enable caching of OCSP responses, to reduce the load on the Authorized Responders (and themselves). If required, also

Trusted Responders may operate in Transponder mode, just as described in section 2.2.1., but this requires configuring the OCSP clients to trust these Responders.

3.5.2.3.OCSP Relying Parties

Relying parties (OCSP clients) must be provided with a configuration capability (like the policy file proposed in section 6.2) sufficient to take advantage of the discovery mechanisms for OCSP Responders, but always meeting local security and reliability requirements.

OCSP overhead is a critical issue to take into consideration for relying parties, therefore a minimum set of provisions should be taken to balance security and performance. Let us mention for example that OCSP clients should not make use of a nonce in their requests as it becomes critical for enabling OCSP caches. In section 6.2.3 we will revisit the OCSP performance topic by showing the results of our research.

3.6 Open Issues

In this chapter have been presented the results of our research on deploying the Online Certificate Status Protocol (OCSP) into the Computational Grid. We have reviewed the main challenges that seem to be delaying OCSP adoption, but also have detailed a set of recommendations towards easing this goal. There is still a lot of work to do with the Grid community to push the use of OCSP as an alternative to CRLs and few other static mechanisms being used in current installations, but prototypes like the one developed during this research (Chapter 6) may be useful for testing purposes among system administrators, developers and Grid users en general.

Despite this initial effort on implementing a validation infrastructure based on Grid-OCSP, there are still open questions about interoperability issues in these multi-authoritative environments. In particular it is easy to notice that OCSP (and this is true also for other validation protocols) can only give information about a Grid user credential's status, but nevertheless about its assurance or security level. This information is required to build the proposed Extended Path Validation process.

At the beginning of this thesis document we justified the importance of applying a security evaluation for trust management purposes, beyond a binary valid/invalid reply. But, how to perform this evaluation? A first step towards finding the "security level" associated with a Grid identity (in particular with an end-entity certificate), may consists in applying an evaluation methodology able to unambiguously quantify it.

In analogy with the trust model implemented by the Policy Management Authorities (section 2.3.3), a suitable approach could focus in evaluating the issuing Certification Policy and then associating the obtained security level to the end-entity certificate's status being requested to the OCSP Responder. Afterwards, an enhanced Grid-OCSP client should be able to extract this information from the OCSP Response to perform a comprehensive validation decision.

The next chapter focuses in explaining an investigated evaluation methodology, which after some subtle extensions, was found to be quite useful to obtain a numeric value representing a Grid-PKI security level. This technique is called, the Reference Evaluation Methodology or REM.

CHAPTER 4 STATIC POLICY EVALUATION: A CONTRIBUTION TO ACCREDIT GRID PKIS

4.1 Motivation

In the last years, security research activities in the computational Grid context have been focused on the definition of infrastructures that enable Single Sign On (SSO) mechanisms to access distributed resources that belong to, and are administrated by, different and independent domains. SSO mechanisms allow an end user to authenticate upon his first access to a resource and let him use those verified credentials for the following accesses to resources that belong to different machines in the same Virtual Organization (VO). From a security point of view, this problem is faced first of all by adopting Authentication and Authorization Infrastructures (AAI), implemented in different ways that include organizational and technical procedures described by policies, and security mechanisms enforcing them. Commonly used AAI were reviewed in section 2.1

Nowadays it is a common approach to demand authentication-related operations to external Authorities acting as Trusted Third Parties (TTP). Specifically in Grid contexts, available solutions are based on the adoption of Public Key Infrastructures that enable the implementation of authentication and authorization mechanisms on the basis of X.509 digital certificates issued and signed by Certification Authorities (CA).

The VOs require the use of certificates that are trusted by all the involved parties, each one of them issued by their home institution's Certificate Authority. These cryptographic credentials are the basis not only of entities authentication (end-users and machines), but also of job delegation mechanisms and even of further authorization processes.

Wide deployment of VOs and Grid-PKIs required the adoption of Policy Management Authorities (PMAs) to establish explicit trust relationships among these (section 2.3.3 introduced basic concepts about PMAs). Even though all PMA's members Grid CAs must pass the accreditation process, not all of them accomplish the respective Authentication Profile on the same level. As shown in [LoA] this information is very important to build trust relationships among Grid installations, however to date there is no automatic way to obtain it.

The contribution presented in the following sections is focused on the problem of building a technique aimed for PMAs to automatically perform the Grid-PKI accreditation process by evaluating its Certification Policy and obtaining useful information from it, for example a numeric security level. This information can be used later to assess Certification Authorities in improving their security features. The main components for validating digital certificates under these premises are:

- An evaluation methodology (to be introduced in section 4.3), which is based on the formalization of the CA's Certification Policy to *i)* determine if a CA is compliant with the Authentication Profile defined by the PMA and *ii)* by defining a security metric to quantitatively compute a Global Security Level (GSL) representing how much better the CP's security provisions are when compared against the PMA's Authentication Profile.
- A Grid Validation Infrastructure –extended from the system introduced in chapter 3- able to convey the status and associated GSL of any end-entity certificate through the Online Certificate Status Protocol previously explained in section 2.2.1.

The security level obtained from the CP evaluation presented in this chapter, can then be conveyed and used by relying parties to enhance the process of validating any client submitting a job to the computational Grid. This topic will be further explained in chapters 5 and 6, where the last contribution of the present thesis is covered: the Policy and OCSP based Interoperability System.

4.2 Quantifying the PMA's accreditation process

As mentioned in section 2.3.3, the PMA's Authentication Profile document describes the minimum set of requirements imposed on Certification Authorities (CA) that traditionally issue long-term X.509 credentials to end-entities, who will themselves posses and control their key pair and activation data. These CAs act as an independent Trusted Third Party for both subscribers and relying parties within the Grid infrastructure.

In general, any CA requesting membership to a PMA must present its Certification Policy (CP) document which will be in turn evaluated by experts against a set of rules or minimum security provisions defined into the PMA's "Authentication Profile", thus determining whether it is fulfilled or not. Obviously a further auditing process could determine if in practice the CP itself is also being satisfied by the PKI.

According to practical experience in this field, it is possible for all the Grid-PKI members of a certain PMA to not only fully satisfy the appropriate "Authentication Profile", but also to do so in different levels. Let us show the following example to clarify this point of view:

Example 1. Suppose that both *PKIa* and *PKIb* are part of the same PMA whose Authentication Profile document requires an end-entities' private key minimum length of 1024 bits. On the other hand, *PKIa*'s CP states that its users own private keys of at least 1024 bits while *PKIb*'s CP states a minimum of 2048 bits. Even though both PKIs fulfil the PMA minimum requirement, a certificate issued by *PKIa* may have a private key smaller than one issued by *PKIb*. In a broad sense of the word, it could mean that for this particular provision *PKIa* is less secure than *PKIb*. If the rest of the CP's provisions for both PKIs are exactly the same, then the global security level of *PKIa* should be less than *PKIb*'s. Under these conditions a Grid resource owner may decide to give access only to users with a certificate issued by *PKIb*.

To perform validation decisions as the one presented in the previous example, it is necessary to evaluate the *distance* between a particular Grid-CA's Certification Policy and a reference PMA's Authentication Profile. Quantifying this "security level" may be useful also for relying parties willing to take finer-grained authorization decisions.

From the research presented in this thesis, the Reference Evaluation Methodology or simply REM [REM1] resulted as an appropriate mechanism for quantifying the security level associated with a Grid-PKI. The proposed evaluation methodology is general enough to be applied in all cases in which cooperation is possible only after explicit agreements among parties, via for example a Trusted Third Party (i.e. the Policy Management Authority).

The following section explains the basic concepts of this evaluation methodology.

4.3 The Reference Evaluation Methodology

In [REM1], [REM2], [REM3] and [REM4] were proposed the Reference Evaluation Methodology to evaluate and compare different security policies, quantifying both their *Global Security Level* (GSL) and their compliance to a declared SLA –Service Level Agreement-.

In particular the REM is made of three different components:

- The policy formalization: defines how to express a security policy in a rigorous way.

- The evaluation technique: the process to evaluate a previously formalized policy. Once defined an evaluation technique, it can be used to define a security metrics in order to state the security level of the infrastructure.
- The reference levels: denotes how to get a security level relative to a minimum policy used as reference

In this section is provided a high level description of the evaluation methodology, which consists of the following phases:

1. Policy Structuring (section 4.3.1): consists in finding a common structure for certificate policies that are usually expressed by means of a natural language.
2. Policy Formalization (section 4.3.2): a formal representation is obtained from the structured policy.
3. Policy Evaluation (section 4.3.3): a distance function is defined to evaluate and compare policies.

4.3.1. Policy Structuring

The Policy Structuring phase consists of three steps and its goal is to define a not ambiguous policy framework and then use it to structure the Certificate Policy to evaluate. For this research was decided to use the "Global Grid Forum Certification Policy Model" [GFD16] as the document upon which the REM's template could be built. This was not only because it is a common practice for the Grid community to use it for writing their own CPs, but also because all of the provisions from EUGridPMA's Authentication Profile [GridAP] can be easily mapped to it.
Parsing the GGF's model CP into the form required by the REM is a tough process as it requires not only clearly identifying each one of the possible security-related provisions, but also organizing and placing them into the leaves of the formalization tree. The first levels of this tree are precisely those defined in the RFC3647 [RFC3647]; this document is actually the most important reference to a Certification Policy structure, so for the first level we have allocated the following sections:

1. *Introduction,*
2. *General Provisions,*
3. *Identification and Authentication,*
4. *Operational Requirements,*
5. *Physical, Procedural and Personnel Security,*
6. *Technical Security Control,*
7. *Certificate and Certificate Revocation List (CRL) Profiles,*
8. *Specification Administration.*

For the Second level of the tree we describe all the details regarding every macro-provisions (first level) by stating objects that despite their complexity, bring more security bounded information. Let us take for example the "Technical Security Control" provision which includes: "Key Pair Generation", "Private Key Protection", "Other Aspects of Key Pair Management", "Activation Data", "Computer Security Controls", "Life Cycle Technical Control", "Network Security Control" and "Cryptographic module engineering controls".
After the complete tree has been generated each node must be refined by introducing enumerative and ordered data types, each representing a *security provision* (typing step). The output of the typing step is a policy representation *P* that is used to represent any Certificate Policy (translation phase). Translation of individual Certificate Policies to evaluate is needed to map them into the formalized *P*. Up to date, almost all the steps

of this phase are not automatable; nevertheless, while the tree generation and typing steps could be performed just once, the translation step needs to be performed once for each policy to represented by *P*. On the other hand it is worth highlighting that in most of the cases Certificate Policies do not change drastically during a PKI lifetime.
At the end of the Structuring phase, each policy is represented by a non numeric vector *P* of not heterogeneous provisions.
The fully structured CP template for Grid-PKIs is presented in appendix A (notice that in this case *n = 166*). This template will be widely used for the rest of this chapter.

Example 2. Based on Appendix A let us take a couple of the provisions from the "Technical security controls" section:

Key pair generation and installation = {generatedByCA, generatedByEntity}
Asymmetric key sizes = {512, 1024, 2048}

A wide number of Grid-PKIs have published policies supporting different security levels, for this reason, before continuing, we need to introduce the definition of "Global Security Level" associated to the Certificate Policy and "Local Security Level" associated to the single provisions of the policies.
The *Global Security Level (GSL)* is representative of the class of the Certificates that a CA could issue. A CA defines different classes of certificates that are strictly related to the *assurance level* the certificate is able to guarantee and consequently to the different applications the certificates are best suited for. For example if certain CA issues three different classes of certificates (i.e. Silver, Gold and Platinum), then it must publish three different Certificate Policies, one for each class.
The *Local Security Level (LSL)* is strictly related to the assurance level that each provision is able to guarantee to the users. To formalize this concept it is necessary to introduce some formal notations.
Let *P* be a policy consisting of a finite positive number *n* of provisions K_i *(i = 1..n)*. We define each provision K_i to be a data type, i.e. a set of *m* different values: $K_i = \{a_{i,1}, a_{i,2} \cdots a_{i,m}\}$; each value implies a different security level.
To introduce the concept of local security levels, a total order relation "$<_i$" must be defined on K_i. An order relation could already exist (for example if K_i is a numeric set), otherwise the elements of K_i must be ordered according to some criteria established by the policymaker. Based on example 2, we show in example 3 the corresponding local security levels:

Example 3. The following order relation shows that the greatest LSL is achieved if the End-Entity generates its own key pair:

Key pair generation and installation = { $L_0 < L_1$ }

Where: L_0=*{generatedByCA},* L_1 =*{generatedByEntity}*
And for the key sizes:

Asymmetric key sizes = {512 < 1024 < 2048}

Each provision is now represented as an enumerative and ordered data-type; the policy space "*P*" is then defined as the vector product of all *n* provisions K_i i.e. $P = K_1 \times K_2 \times \ldots \times K_n$.

The provisions shown in the previous examples can not be evaluated directly; they need to be transformed into a *numeric* (normalization) and *homogeneous* (clusterization) vector first. The definition of this vector is the goal of the "Formalization" phase.

4.3.2. Policy Formalization

Policies may be expressed in both, a formal or in an informal way. Despite its advantages for an automatic processing a formalized policy is more difficult to manage and understand, so often security policies are simply expressed into a semi-formal way, but then again even though their structure is easier to understand -even for non technical people-, they are still difficult to be automated. A formalized policy instance expresses in a rigorous way, who, how and where security provisions will be applied. The way in which a policy is formalized depends strongly on the evaluation technique intended to adopt.

In this phase will be transformed the Policy Space *P* into an homogeneous space *PS* and then, defined a distance criterion among policies to build a *Metric Policy Space*. This Metric Policy Space is formally defined as the pair *MPS = (PS, d)* where *PS* is an homogeneous Policy Space and *d* is a metric function. Given two policies $P, Q \in PS$, *d(P,Q)* is the *distance* from *P* to *Q*. The next step is to define the mentioned homogeneous space; this task consists of two different steps: *i)* normalization and *ii)* clusterization.

The normalization step aims at assigning a "numerical value" to each provision while the goal of the clusterization step is to associate a comparable security mean to each provision by means of the formal definition of absolute LSLs. Only after clusterization, the policy space can be considered homogeneous.

For the normalization, due to the total order relation $<_i$ it is possible to associate a progressive numerical value to each value in K_i, normalized with respect to the number of values *(m)* that the element itself can assume. In the following example we associate the numeric value *i/m* to the i^{th} element of the provision.

Example 4. For the normalization phase:

Key pair generation and installation = { generatedByCA < generatedByEntity }

generatedByCA := 1/2

generatedByEntity := 2/2

Asymmetric key sizes = {512 < 1024 < 2048}

512 := 1/3

1024 := 2/2

2048 := 3/3

After the normalization, each policy *P* can be represented by a numeric vector, nevertheless, *P* is not homogeneous yet. Each element of the vector is, in fact, representative of a different provision and its numerical value is not comparable with the values associated to the other provisions, representing very different informative contents; i.e. the comparison among numeric values associated to different elements is meaningless. At the aim of giving to different provisions a comparable security value, it is necessary to apply *clusterization*.

For the clusterization, the provisions will be literally grouped into clusters according to the Local Security Level assigned to each cluster. At this aim it is suggested to adopt a clusterization technique based on a family of transformations (the *F-transformations*) to map the normalized provisions into a set of predefined LSLs.

Formally speaking, let *P* be a policy and $\bar{f}_i$ be the transformation: $\bar{f}_i : K_i \to L$, where K_i is a provision of *P* and *L* is the set of the values associated to the LSLs ($L = \{L_0, L_1 \cdots L_z\}$). According to this $\bar{f}_i$-transformation, each instance of the provision K_i will be assigned to a local security level.
Let us assume that *L* consists of four elements (L_0, L_1, L_2, L_3), this choice is a realistic one since it is the case of several examples of PKIs (i.e. with certificate classes Copper, Silver, Gold and Platinum). Let *m* be the number of instances of the provision K_i, it is possible to distinguish three different cases: *m = 4*, *m < 4* and *m > 4*. Then the suggested transformations are:

- If *m = 4*:

  $$\bar{f}_i(a_j) = l_j \;\; (\forall i = 1 \ldots n, \forall j = 1 \ldots m).$$

- If *m < 4*:

  $$\bar{f}_i(a_1) = l_1,\, \bar{f}_i(a_m) = l_4,\, \bar{f}_i(a_j) = \{l_2, l_3\} \;\; (i = 1 \ldots n, j = 2 \ldots (m-1)).$$

- If *m = 2p* and *p > 2*:

  $$\bar{f}_i(a_j) = \begin{cases} l_1 \Leftrightarrow j = 1 \\ l_4 \Leftrightarrow j = m \\ l_2 \Leftrightarrow j = 2 \ldots m/2 \\ l_3 \Leftrightarrow j = (m/2+1) \ldots (m-1) \end{cases}.$$

- If *m = 2p+1* and *p* ≥ *2*:

  $$\bar{f}_i(a_j) = \begin{cases} l_1 \Leftrightarrow j = 1 \\ l_4 \Leftrightarrow j = m \\ l_2 \Leftrightarrow j = 2 \ldots (m+1)/2 \\ l_3 \Leftrightarrow j = ((m+1)/2+1) \ldots (m-1) \end{cases}.$$

Example 5. Applying the clusterization criteria to the provisions from Example 4:

Key pair generation and installation = { generatedByCA < generatedByEntity }

generatedByCA := 1/2 $\to L_0$

generatedByEntity := 2/2 $\to L_4$

Asymmetric key sizes = {512 < 1024 < 2048}

512 := 1/3 $\to L_0$

1024 := 2/2 $\to (L_2, L_3)$ Notice that in this case $\bar{f}_i$ maps to two LSLs, because *it is not a function*.

2048 := 3/3 $\to L_4$

Once the Certificate Policy has been structured and formalized, it is possible to proceed directly to the evaluation just as explained in the following section.

4.3.3. Policy Evaluation

Different evaluation techniques represent and characterize the security level associated to a security policy in different ways, for example with a numerical value [OSSTMM], a fuzzy number [REM2] or a verbal judgment representing its security level.
Each *PS* (Certificate Policy structured, normalized and clusterized) can be represented by a *nx4* matrix as follows:

- *n* rows represent the structured CP's provisions.
- *Four* columns represent the possible LSLs.
- Each element a_{ij} *(i = 1..n, j = 1..4)* of the matrix is defined as follows:

$$a_{ij} = \begin{cases} 1, & j \le x \\ 0, & j > x \end{cases}$$

where $x \in [1,4]$ and $f_i(K_i) = l_x$

Example 6. For a particular CP the following values are specified:

Key pair generation and installation = { generatedByEntity } and

Asymmetric key sizes = {1024}

Therefore the following rows will be its representation into the correspondent matrix:

Key pair generation and installation = {1,1,1,1}

Asymmetric key sizes = {1,1,1,0}

For the Certificate Policy evaluation a distance criteria will be used to define the Global Security Level –GSL- as the *Euclidean distance among matrices*, that is:

$$d(A,B) = \sqrt{(\sigma(A-B,A-B))}\,.$$

Where:

$$\sigma(A-B,A-B) = Tr((A-B)(A-B)^T).$$

And A^T is the transport of matrix A, while *Tr(A)* is the trace of *A* (the sum of all the elements on its main diagonal).

Example 7. To demonstrate that the distance just defined represents the distance between the policies being evaluated, below are shown 3 policies (*P*, *X* and *Y*) with 10 provisions each. *P* is then compared against *X* and *Y* -both are globally stronger than *P* since they have several provisions with a higher Local Security Level-.

$$P = \begin{vmatrix} 1&1&0&0\\1&1&0&0\\1&0&0&0\\1&0&0&0\\1&0&0&0\\1&0&0&0\\1&0&0&0\\1&0&0&0\\1&1&0&0\\1&0&0&0 \end{vmatrix} \quad X = \begin{vmatrix} 1&1&1&0\\1&1&1&0\\1&1&0&0\\1&0&0&0\\1&0&0&0\\1&1&0&0\\1&0&0&0\\1&1&0&0\\1&0&0&0\\1&0&0&0 \end{vmatrix} \quad Y = \begin{vmatrix} 1&1&1&1\\1&1&1&1\\1&1&1&0\\1&1&0&0\\1&1&1&0\\1&1&1&1\\1&1&1&1\\1&1&0&0\\1&1&0&0\\1&1&1&1 \end{vmatrix}$$

The following conclusions can be drawn from the previous figure:

- Just looking at the levels of the single provisions, *X* is a policy that appears stronger than *P*. First we calculate:

$$Tr((Y-P)(Y-P)^T) = 6$$

So the distance between *X* and *P* is: *d*= 2.45

That mirrors the fact that *X* is just a little stronger than *P*.

- *Y* is a policy that appears stronger than *X* and much stronger than *P:*

$$Tr((Y-P)(Y-P)^T) = 19$$

In this case the distance between *Y* and *P* is: *d= 4.36* and this shows the evident security difference between the both policies.

The previous example shows that it is very simple to evaluate the distance between formalized and structured Certificate Policies, once they have been represented as a matrix. This distance criterion will be adopted to define a metric function for the Global Security Level just as explained next.

4.3.4. Global Security Level

The last component of the REM to compute a Global Security Level (GSL) is the set of reference security levels from a minimum policy, which can be used as a scale for the numerical evaluation of trustability. When these references are not available, the REM could be used for direct numeric comparison among two or more policies as explained in the previous sections.

At this point, to define the metric function for the GSL we must represent the reference levels according to the REM's procedure and then proceed to evaluate the target policy against them. So, we first evaluate the distances among the different references (denoted as $REFL_i$) and the origin of the metric space (denoted as Ø) which is represented as $d_{REFLi,\phi}$

Finally, the security metric function to evaluate the Global Security Level (GSL, for short) associated to a target policy P_x will be:

$$Lp_x = \begin{cases} L_0 & \text{iff } d_{x0} \le d_{10} \\ L_1 & \text{iff } d_{10} < d_{x0} < d_{20} \\ L_2 & \text{iff } d_{20} < d_{x0} < d_{30} \\ L_3 & \text{iff } d_{30} < d_{x0} < d_{40} \\ L_4 & \text{iff } d_{40} \le d_{x0} \end{cases}.$$

Where Lp_x is the Global Security Level associated to P_x and $d_{i,0} = d_{REFLi,\phi}$ are the distances among the references and the origin of the metric space. In this particular case we have established five security levels (from L_0 to L_4), but this decision may change according to the scenario being evaluated.

Being an aggregated value, the GSL has a critical limitation: it is not able to take into account single provisions which could be more critical than others. To face such problems the REM can be enhanced by applying weights to provisions and introducing a constraint-vector, which could guarantee that during the evaluation even if the aggregated evaluation gives a high GSL, some critical provisions do not go under a desired threshold. To take in consideration the weights in the matrix representation we can multiply each elements of a row for the corresponding weight: let $\beta \in [0,1]$ be a weight, then the row $\beta \cdot (1,1,0,0) = (\beta, \beta, 0, 0)$.

The rest of this chapter presents a proposal to use the REM technique for improving the Grid-OCSP infrastructure introduced in Chapter 3.

4.4 A REM-enhanced Grid-OCSP Validation Infrastructure

To perform a comprehensive certificate validation process, it is necessary to query in real-time its status and evaluate the security level of the issuing CA. For the first task the proposed system uses a Grid Validation System able to retrieve the status of a digital certificate through OCSP in a CA federation; this system (introduced in section 2.2.3) was developed as part of our research (section 6.2 presents all the details behind its implementation).

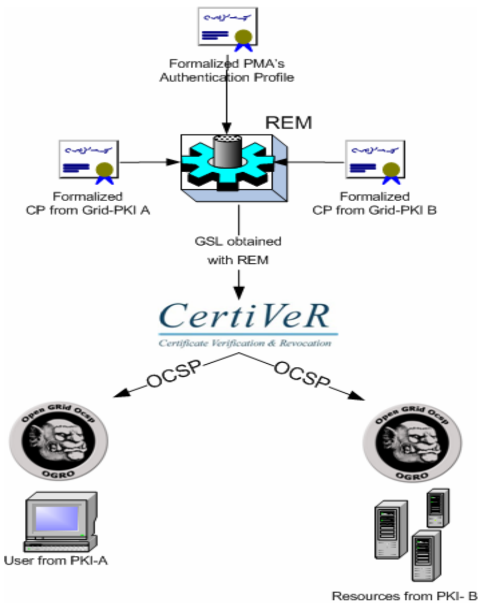

Formalized PMA's Authentication Profile
REM
Formalized CP from Grid-PKI A
Formalized CP from Grid-PKI B
GSL obtained with REM
CertiVeR
Certificate Verification & Revocation
OCSP
OCSP
Open GRid Ocsp
OGRO
Open GRid Ocsp
OGRO
User from PKI-A
Resources from PKI- B

Fig. 16. The extended Grid-OCSP Validation Architecture.

On the other hand, for the security evaluation of a CA this proposal applies the Reference Evaluation Methodology for testing compliance with a PMA's Authentication Profile and, quantitatively computing a Global Security Level. This information can be used afterwards for a comprehensive Path Validation process.
Figure 16 shows the overall validation infrastructure proposed by this research, which enhances the Grid-OCSP Architecture from chapter 3 by introducing the REM evaluation. As a proof of concept, the following section will demonstrate the usefulness of the REM's methodology to evaluate a set of Certification Authorities from EUGridPMA.

4.5 Static Certification Policy evaluation of Grid PKIs

The goal of this section is to show the practical utilization of the proposed validation architecture by using the policy matrix obtained in the previous section and the Reference Evaluation Methodology, to quantify a Global Security Level –GSL- for the Certification Policies corresponding to the Grid-PKIs members of EUGridPMA.
With this aim all of the Certification Policy documents must be "translated" into a policy matrix instance. This translation process is *i)* asynchronous (can be performed at anytime prior to the Grid user validation), *ii)* semi-automatic (at this time CP's parsing is performed by a human) and *iii)* repeated every time the Certification Policy changes (which in practice happens rarely).
When translating a CP, what really happens is a parsing and mapping of each provision from this document to the policy matrix template shown in Appendix A. This is very easy to perform (at least for a human) if we consider that most Grid-PKIs CPs are an instance of the GGF's CP model [GFD16] and therefore its provisions may be directly mapped to the template.
In our particular case the first CP to evaluate was the EUGridPMA's Authentication Profile [GridAP], because its security level will determine the minimum reference GSL for the policy space. It is expected that any other Grid-PKI's GSL should be equal or

greater than this value (considering that all the evaluated CPs are only from EUGridPMA members).
When evaluating a CP it is important to highlight two special cases: *i)* those provisions found only into the CP under evaluation and not into the template will be omitted from the evaluation process and, *ii)* those present in the template, but not in the policy, will be mapped to the worst LSL (in the policy matrix this means a row with zero in every column) and shall be considered *notStipulated* clauses. These two choices were motivated because the PMA's minimum requirements document defines by itself which provisions must be considered in the evaluation process to accredit a new Grid-PKI, thus with the REM methodology we have only extrapolated this criteria.
Once the CP has been translated, an API implementing the REM technique was used to obtain the corresponding GSL. In its more general form this API (written in Java) receives three vectors of integers as input parameters: a first one containing the values for each provision of the CP to evaluate, a second one with the values of the CP to use as a reference (i.e. the PMA's authentication profile) and finally, a third vector with the allowed cardinality for each provision. The output is a double representing the relative distance among both Certification Policies.
Applying the methodology and API explained in this chapter was obtained the GSL for each one of the Certification Authorities members of the EUGridPMA [EUGridPMA]. Figure 17 compares these levels in a graphical way for a sample composed of the EUGridPMA's minimum Authentication Profile [GridAP] and the Certification Policies from IRIS Grid CA [IrisCP], US Department of Energy Grids CA [DoECP], CERN CA [CernCP] and INFN CA [InfnCP].

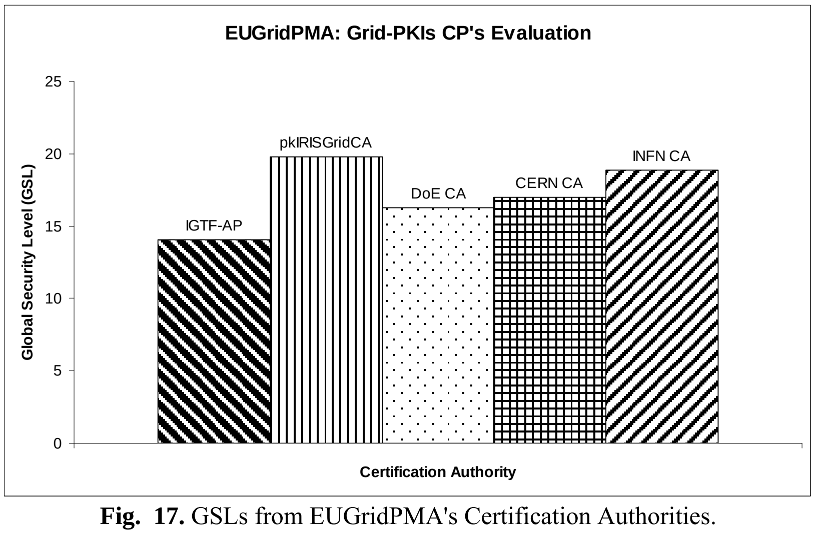


Fig. 17. GSLs from EUGridPMA's Certification Authorities.

First of all, it is easy to note that every CA had a GSL greater than the reference (in fact the distance between CPs was quite large). This was an expected result as every CA complies with EUGridPMA's minimum requirements (that is a GSL greater than IGTF-AP's which GSL=14.03). However also was found that -just as stated at the beginning of this chapter- these GSLs differ from one member CA to another, therefore this new information may give the opportunity for a Grid-PKI to dynamically decide to interoperate only with those institutions whose GSL is equal or greater than its own. Let us take the following scenario as an example.

Example 8. INFN Certificate Authority has a GSL=18.86, so it is possible to believe that they could choose to work only in some projects with end-entities coming from the IRISGrid CA (GSL=19.77). Moreover they also could establish their own minimum requirements for some project -in other words a new CP template- and then work only with those institutions fulfilling this new set of provisions.

As already said, the GSL is an aggregated value and by itself can not give information about specific provisions of the Certification Policy; this could be a problem in those contexts where it is mandatory that all provisions respect the minimal requirements.
Fortunately, thanks to the proposed formalization process it is even possible to evaluate each section of the Certification Policy to obtain partial-GSLs that may allow the relying party to perform a finer validation decision. Table 2 shows -for EUGridPMA's chosen set of CAs- the partial GSLs from each one of the sections composing the Certification Policy template. In this table it is interesting to note the degree of detail that can be obtained when further comparing these Grid-PKIs; it is particularly noticeable the fact that even though all of them have an overall GSL greater than EUGridPMA's, there are some CA's partial-GSLs that do not follow this criteria (i.e. for the "Technical security controls" section: $partialGSL_{pkIRISGrid} < partialGSL_{IGTF-AP}$).

Table 2. Partial GSLs from EUGridPMA's CAs

CP Section	IGTF-AP	pkIRISGrid	DoE	CERN	INFN
Introduction	5.00	5.57	5.48	5.66	5.39
General Provisions	7.00	10.05	8.37	9.06	9.43
Identification & Authentication	4.90	5.10	4.69	5.29	5.92
Operation Requirements	5.39	10.00	6.93	7.87	9.80
Physical, procedural and personnel security	1.73	7.35	3.32	2.45	4.69
Technical security controls	5.83	5.48	5.48	6.00	5.92
CA Certificates	3.87	5.39	5.66	5.00	5.83
Certificate and CRL profiles	2.00	2.83	3.46	3.46	2.83
Administration of Specifications	3.16	2.45	2.45	0.00	1.41
Others	2.00	2.45	2.00	2.45	2.45

Apart from using this information for the validation process, it is feasible to think that it can be also very useful for the PMAs to assess Certification Authorities (members or applicants) in better refining or correcting their security policies by analyzing very specific subsets of provisions.
In general, which may be the finest-grained GSL that can be obtained from our methodology? The answer is simple, as it is feasible to get the *GSL for each one of the individual provisions* from the Certification Policy being analyzed. Afterwards this set of GSLs can be used by both the PMA -to accurately assess the CAs in improving weak provisions- and the relying party itself –to take very granular validation decisions according to a predefined template-.
A suitable way to graphically represent and compare the GSL for individual provisions is through a Kiviat Diagram, where each provision is represented by an axis with one division for each possible LSL – in this case, four-. In this way, given an axis representing a particular provision, we can depict the LSLs corresponding to one or

more Certification Policies, which facilitates their comparison at this specific level of detail.
Let us take for example figure 18, where a Kiviat diagram -representing the CP provisions related with the identification and authentication features- contains the LSLs for the IRIS Grid CA and INFN CA. It is possible to notice that for provisions like "Need for names to be meaningful", the EUGridPMA's minimum requirement (IGTF-AP graph) is fulfilled by the CAs. Also there are some others like "Rekey after Revocation" where the minimum requirement is also fulfilled, but with a different LSL by each one of the CAs being evaluated.
On the other hand, a third group of provisions has been identified (i.e. "Authentication of organization identity") in which the EUGridPMA's minimum requirement (face to face identification with the Registration Authority) is not fulfilled by the IRIS Grid CA.

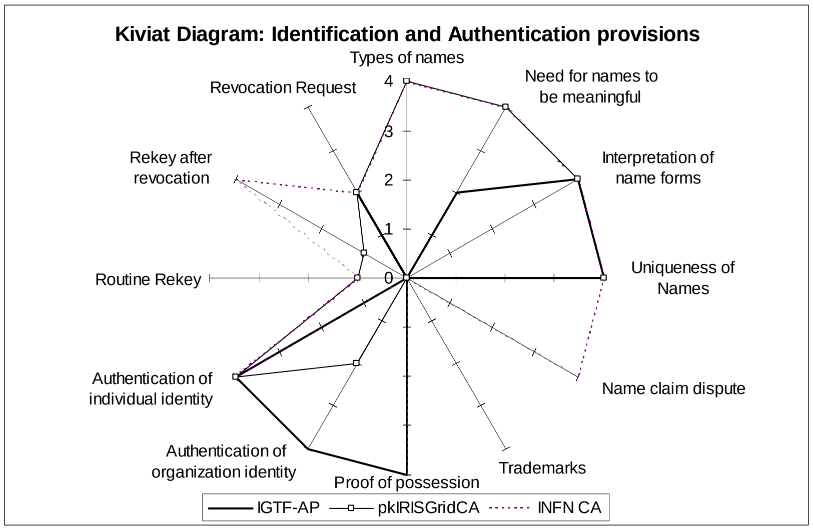


Fig. 18. Kiviat Diagram for CP's Identification and Authentication provisions.

The API implemented by this research is also capable of aiding a PMA to carry out automatically the assessment of individual provisions, just by subtracting the matrices representing the Certification Policies being evaluated (rows with a negative number mean a provision that should be enhanced).

4.6 Open Issues

This chapter presented a contribution of our research towards implementing a "static" credential validation process for the computational Grid. As a way to ease interoperability among different Grid-PKIs the Reference Evaluation Methodology (REM) has been adopted and extended to cope with the challenges imposed by these distributed environments, and as a proof of concept it was applied to obtain the security level associated with Certification Authorities members of the IGTF.
The results obtained with the REM have proved quite useful also for assessing a CA that whishes to become part of a certain PMA, because it is possible to identify the individual provisions of the evaluated CP that must be improved in order to achieve the minimum security requirements of an Authentication Profile.
Contrary to the current PMA accreditation process, which is mostly manual, our proposal can be fully automated as shown by the Java API developed as a proof of concept associated with this research.
Even though in section 4.4 was introduced the notion of a validation infrastructure that uses the Grid-OCSP system and the extended REM technique, there was still the need to

investigate the issues related with its adoption into the Computation Grid as envisioned at the beginning of this thesis, in particular when referring to the idea of an extended and "dynamic" validation process able to ease the interoperability issues of Virtual Organizations in real-world installations. The following chapter presents the methodological approach for performing the Extended Path Validation process though a dynamic validation infrastructure, called the *Policy and OCSP based Interoperability System* (POIS).

CHAPTER 5 DYNAMIC POLICY EVALUATION: A CONTRIBUTION TO EXTENDED PATH VALIDATION

5.1 Motivation

As Grid technology has become widely adopted, so the number of Virtual Organizations has grown. This represents one of the biggest interoperability problems that could arise among all Grid users and therefore one of the major security challenges to be faced before building a wide distributed infrastructure that allows the cooperation of existing Grid installations. From our point of view this problem is related to the definition of a distributed validation infrastructure, able to guarantee a secure degree of interoperability among all the involved Grid-Certification Authorities.

Currently there are two state of the art approaches that provide interoperability between different security domains using PKI-based technologies:

- Involved CAs explicitly build a trusted domain, defining a new CA hierarchy through cross certification techniques. In this case each CA explicitly trusts the others and therefore is able to accept their certificates.
- Involved CAs do not build an explicit trusted domain, but interoperate through a "federation": any CA belonging to the federation implicitly trusts the others thanks to the definition of a well-established policy-based framework.

Even if the explicit trusted domain (first approach) is an attractive solution, it is not always possible to implement in Grid environments, because of the required agreements between the involved organizations, administrative overheads and technical problems that arise with current software (in fact this is the case of the Globus Toolkit [Globus]).

For the computational Grid, the second of the aforementioned options (building a Federation of CAs) has been the most suitable solution for real-world projects so far. At this aim, the Policy Management Authorities (see section 2.3.3) have established a minimum set of requirements and best practices for Grid PKIs willing to join its federation, these are called the PMA's Authentication Profile. In summary, Policy Management Authorities represent "Federations of Grid PKIs" whose CA members are required to accomplish minimum levels of security through an accreditation process.

Moreover, in chapter 4 we showed that even though all the Grid CA members of a PMA must fulfill the established authentication profile, not all of them accomplish these minimum requirements on the same level. Despite the importance of this information for building interoperability relationships and for Authentication/Authorization purposes, to date there is no automatic way to obtain this data.

With independence of the interoperability mechanism chosen (explicit trust or CA-federation), any end-entity invoking a Grid Service's operation from the server, activates an authentication process to attest his identity. This process requires validating the end-entity's digital certificate according to the Basic Path Validation procedure described in [RFC3280]. When involved CAs interoperate thanks to explicit trust agreements, only Basic Path Validation is required: cryptographic verifications and status' checks over the involved certificates. State of the art Grid software provides static mechanisms for the Basic Path Validation, where the administrators manually declare a list of accepted CAs, and locally update the respective CRLs. No need to say that this is a cumbersome process.

However, if the involved CAs are part of a Grid-federation and fulfill the authentication profile in different ways, then *Extended Path Validation* is also needed: Basic Path Validation enhanced with a policy mapping process that evaluates and compares the involved CAs' Certificate Policies, therefore asserting that they are compliant with a particular Authentication Profile and therefore can interoperate among them.

This chapter explains a proposal to enhance the "static" validation mechanism introduced in Chapter 4; this contribution is the result a joint research on the trust topic performed along with the “Università di Napoli, Federico II” and the “Seconda Università di Napoli”. Furthermore this research proposes in Chapter 6 an architecture for implementing Extended Path Validation in Grid environments, using the Grid-OCSP infrastructure presented in Chapter 3 and the REM technique from Chapter 4.

5.2 Basic path validation with Grid-OCSP

From the security point of view, a typical session with the computational Grid consists on the steps shown in figure 19. These involve the Grid user (End-Entity) using its passphrase to create a proxy certificate from its long-term credential (i.e. with the command *grid-proxy-init*). The user may then use an application to invoke a Grid service’s operation from a container [WSRF]. If Message Level Security is being used for authentication [GT4SEC], then the user’s application would call the corresponding proxy credential to authenticate to the remote host by means of a digitally signed message containing the service invocation.

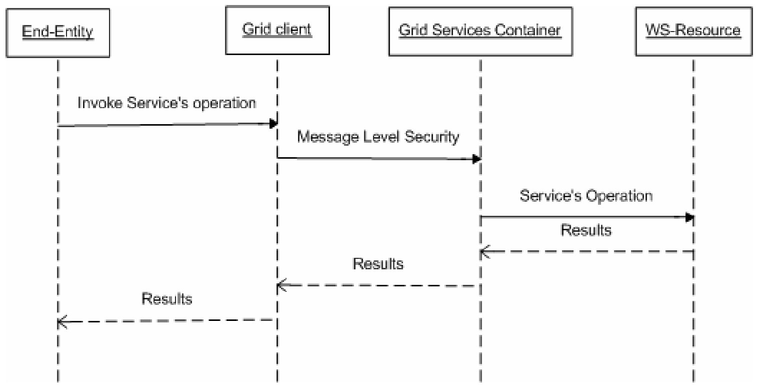


Fig. 19. Invoking a Grid Service from GT4’s Container.

From the Grid resource point of view, to fully perform the authentication process a certificate validation service interface should be defined to be used within the Open Grid Services Architecture [OGSA] implementation to:

1. Parse a certificate and return the desired attribute values –i.e. the validity period-.
2. Perform path validation [RFC3280] on a certificate chain according to the local policy and with local PKI facilities, such as certificate revocation lists (CRLs) or through an online certificate status protocol [RFC2560].
3. Return attribute’s information for generic values, thus allowing the use of different certificate formats or single keys.

Consequently a certificate’s path validation process (step 2 above) must comprise at least the following four phases:

1. Cryptographic verifications over the certificate path (i.e. verifying the digital signature of each certificate).
2. Verifying the validity period of each certificate.
3. Verifying that the first certificate in the chain is a Trust Anchor.
4. Verifying the certificate’s status to ensure that is has not been revoked or suspended.

The process just described is commonly called the *Basic Path Validation*. Modern Grid installations like the Globus Toolkit provide static mechanisms to perform the last two phases of the Basic Path Validation process described above (figure 20):

– The first certificate in the chain is considered a Trust Anchor if it has been stored into the Grid node’s */etc/grid-security/certificates/* directory.

– The certificate's status is retrieved from a locally stored Certificate Revocation List (CRL).

Both processes have an inherent static nature, therefore diverse security problems may arise into the computational Grid.

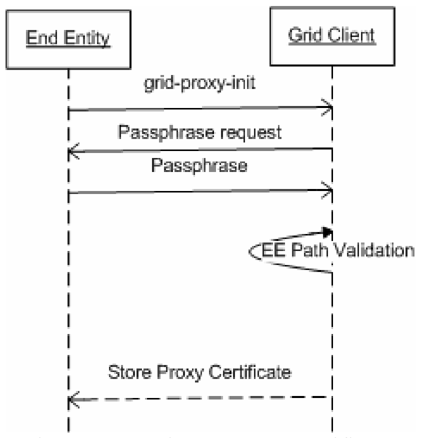


Fig. 20. Creating a Proxy Certificate.

For the purposes of this thesis, the main detected problem was related with interoperability issues between the users and resources belonging to different institutions: the computing resources were in different domains, but the need of cooperation through a Grid environment required sharing them all. Therefore a clear need arose for methodologies, techniques and tools able to build interoperable systems. According to [OGSASEC] the interoperability problem in Grid environments can be subdivided into three levels:

- *Protocol Level*, i.e. the capability of Grid systems to communicate with known and accepted standard protocols.
- *Policy level*, i.e. the capability of each party of the Grid to be able to specify its security policy expressed in a way mutually comprehensible.
- *Identity level*, i.e. the capability of identifying users from one domain to another.

State of the art Grid solutions focus mainly on the first level, accepting the use of SOAP/HTTP protocols as the common platform for system interoperability.

A first approach to enhance the Basic Path Validation process may use the Grid-OCSP explained in Chapter 3. In this case is possible to introduce a high-level OCSP Responder like CertiVeR (section 2.2.3) when *i)* the user creates a proxy certificate (figure 21) or *ii)* when the Grid Services Container is verifying the digital signature (figure 22) from a service's request.

In the first case (figure 21), once Basic Path Validation has taken place, the Grid-OCSP Client sends an OCSP Request to the Responder and the corresponding OCSP Response data is embedded as an X.509 version 3 extension into the proxy certificate itself (the prevalidation mechanism). At this point the proxy certificate is ready to be used.

On the Grid server-side (figure 22) when the user invokes a service's operation from the Grid-OCSP enabled GT4's Container, the OCSP process is triggered after the Basic Path Validation has been performed. If the proxy certificate has prevalidation information, then the OCSP Response data is extracted, verified and processed as explained in the previous section. Notice that the OCSP Responder does not need to be contacted again by the Container, thus enhancing the performance of the validation process.

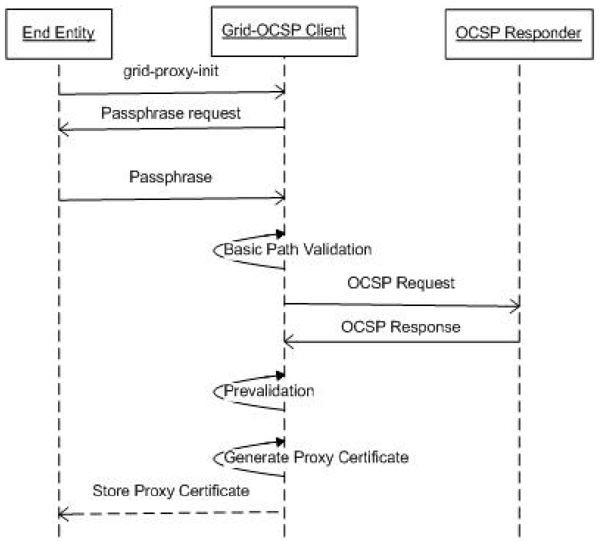


Fig. 21. Using Grid-OCSP and prevalidation when a proxy certificate is created.

The proposal presented in the rest of this chapter focuses on the Identity Level, adopting a policy-based approach to implement an *Extended Path Validation* mechanism beyond Grid-OCSP, just as introduced next.

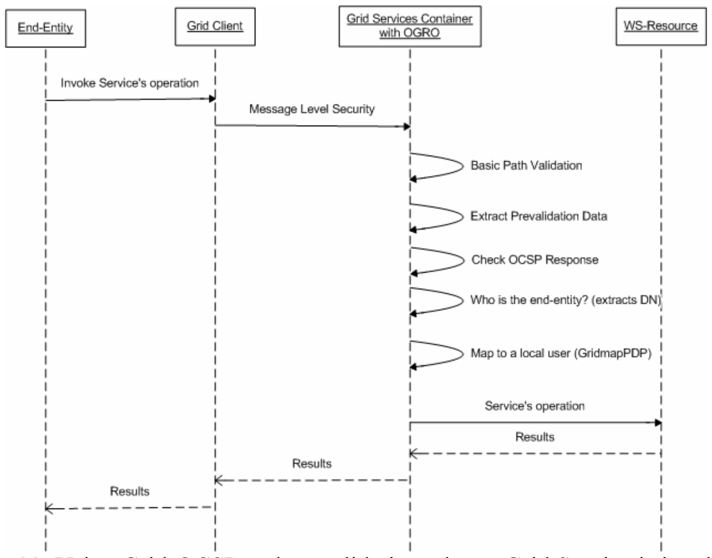

End-Entity
Grid Client
Grid Services Container with OGRO
WS-Resource
Invoke Service's operation
Message Level Security
Basic Path Validation
Extract Prevalidation Data
Check OCSP Response
Who is the end-entity? (extracts DN)
Map to a local user (GridmapPDP)
Service's operation
Results
Results
Results

Fig. 22. Using Grid-OCSP and prevalidation when a Grid Service is invoked.

5.3 Extended Path Validation

The main idea behind the Extended Path Validation mechanism is to define an approach which enables any Grid relying-party to validate in real-time a digital certificate issued by any other CA, even though they do not belong to the same trusted domain (i.e. institution or project). To perform an Extended Path Validation we need:

- A mechanism to validate on-line and near-real time the certificate status of any entity, therefore providing a minimum assurance to the involved parties. This mechanism is the Grid-OCSP system, just as presented in the previous section;
- A methodology to automatically perform the policy comparison and evaluation, to build a dynamic virtual-CA federation where its members fulfil a minimum security level;

As mentioned in section 4.2, most Grid PKIs working together are not completely unknown, but they have been previously accredited by a Policy Management Authority (PMA), which defines a minimum set of security requirements – in the form of an Authentication Profile- that must be accomplished for interoperability reasons. If we measure the degree of compliance of a Grid-PKI's Certification Policy with respect to this Authentication Profile, then it may be possible to build comprehensive trust relationships between those CAs.
In this part of our research we contributed with a methodology for the Extended Path Validation, which guarantees secure interoperability among untrusted domains by *(i)* using Grid-OCSP for retrieving in near real-time the status of any certificate issued by a CA, and *(ii)* evaluating with the extended REM the security level associated with its Certification Policy. The basic functionalities of this *Interoperability System* are explained next.

5.4 The Interoperability System

The goal of the proposed Interoperability System is to enable Extended Path Validation in untrusted Grid domains by building a dynamic federation of CAs. This goal is achieved by evaluating their Certificate Policies with the REM technique. However, in order to have grants about the CA's compliance with its published Certification Policy, there is still the need for the PMA as a Trusted Third Party that acts only as an auditor instead of being also a CP evaluator.
At a coarse grain, the proposed *Interoperability System* (IS) -figure 23- acts as an intermediary between the certificate verifiers (relying parties) and the issuing CAs by managing (retrieving, elaborating and updating) the information required to perform the Extended Path Validation process: the list of accredited CAs, the list of revocation sources and the Certificate Policies.

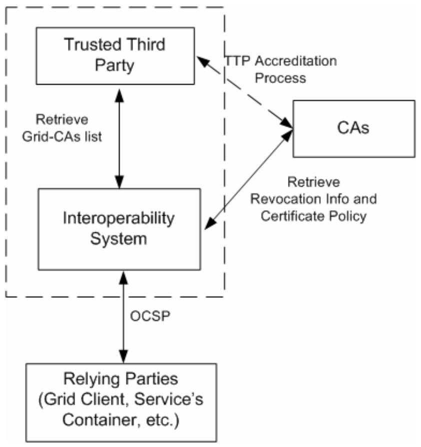

Trusted Third Party
TTP Accreditation Process
CAs
Retrieve Grid-CAs list
Retrieve Revocation Info and Certificate Policy
Interoperability System
OCSP
Relying Parties (Grid Client, Service's Container, etc.)

Fig. 23. Functional blocks of the proposed Interoperability System (IS).

The IS may be collocated with the Trusted Third party and must perform two main tasks:

1. Online validation of the certificates' status and,
2. Evaluation of the issuing CA's security level through its Certification Policy.

Implementation details related with the proposed Interoperability System will be analyzed in chapter 6.

5.5 Open issues

Along this chapter we have presented a Validation Infrastructure for Grid Services, the Interoperability System, which has been designed to implement the Extended Path Validation process. Thanks to this proposal it is feasible to dynamically evaluate an end-entity certificate not only by retrieving its revocation status, but also obtaining a numeric value that represents the security level associated with its Certification Policy. Both features may enable the dynamic building of trust relationships among different Certification Authorities, therefore easing Grid interoperability.

During this research however we found the need to develop a prototype of the proposed system, not only as a proof of concept to backup our ideas, but also to allow feedback from the Grid community and promote its adoption into widely used installations (i.e. the Globus Toolkit). The next Chapter details, from an architectural point of view, the implementation of the proposed Interoperability System.

CHAPTER 6 IMPLEMENTATION AND RESULTS

6.1 Implementing a Grid-OCSP Validation Infrastructure

The Interoperability System introduced in section 5.4 was implemented during our research in two different stages: the Grid-OCSP infrastructure and the REM technique.
In first place was developed and deployed a test Grid-OCSP Validation system able to retrieve the status of a digital certificate through the Online Certificate Status Protocol in a CA federation; this system was built over the CertiVeR high-level OCSP Responder (section 2.2.3) and the novel *Open GRid Ocsp –OGRO-* API to be explained in section 6.2.
In a second stage, for evaluating a CA's security level, we have implemented a Reference Evaluation Methodology API as presented in section 4.3. REM technique's advantages as a tool to ease the PMA's accreditation and assessment processes have been widely highlighted through this work. Section 6.3 uses the developed REM API along with the Grid-OCSP infrastructure to implement our last contribution to the Grid community: the *Policy and OCSP based Interoperability System –POIS-*.

6.2 OGRO: the Open GRid Ocsp middleware for the Globus Toolkit

In the paper [UniAAI] we introduced the basis of an OCSP client for the Globus Toolkit 4 (GT4) able to use CertiVeR (section 2.2.3) both, for proxy certificate's OCSP path validation, and also to request authorization information in OCSP extensions from such service. As a proof of concept this client evolved since then and was published under an open source license with the name of OGRO -Open GRid OCSP– [OGRO].
Besides implementing the relying party's OCSP Requirements (section 3.2), the OGRO API integrates a series of major performance enhancements like prevalidation and second-level caches, which will be explained below.
Other minor contributions were also developed as part of OGRO; take for example the implementation of a solution for validating full certificate chains in GT4 with just one OCSP Request, just as described next:

1. Obtain from the Java Commodity Grid [CoG] classes the full proxy certificate path, beginning with the Root CA certificate and from there down to the last proxy credential (including the End Entity Certificate –EEC-). The Java Commodity Kit or simply CoG, is used at the Java core of GT4.
2. Also from CoG classes obtain the *GSIConstants'* certificate type for each entry from the proxy's path to validate: Proxy (any of *GSI_2_PROXY, LIMITED_PROXY, GSI_3_IMPERSONATION_PROXY, GSI_3_LIMITED_PROXY, GSI_3_INDEPENDENT_PROXY* or *GSI_3_RESTRICTED_PROXY*), CA or EEC.
3. Create one OCSP Request, containing a certificate identifier for each element from the proxy certificate path.
4. Prepare the OCSP Request (i.e. apply nonce and signature as required) and send it to the appropriate OCSP Responder.
5. If successful, then parse the OCSP Response status for each certificate of the original request and process it as required.

A summary of features that make OGRO suitable for Grid environments is shown in table 3. From the features mentioned in this table it is worth highlighting that OGRO introduces a new concept, the *Grid Validation Policy*, which has been defined as a flexible set of relying party's XML rules to configure OCSP behavior in a typical Grid installation. The next section covers the details related with the Grid Validation Policy.

Table 3. Main features in the OGRO client API

Feature	Explanation
Open source.	OGRO API is currently available under an Apache like license.
Java based.	It is 100% Java and as a proof of concept it has been integrated into the GT4 WSRF Java Core and the Java Commodity Grid Kit.
Ease of configuration.	The set of rules mandating its behavior are written in XML and are also highly flexible.
Flexibility of use.	With some minor changes OGRO may be used in other applications (i.e. Web Services). In fact OGRO source distribution includes also a stand-alone OCSP client.
Cryptographic Provider compatibility.	The current version of OGRO uses the Bouncy Castle's libraries as its cryptographic provider, however it can easily be modified to allow for the use of others like IAIK, Cryptix, etc.

6.2.1. Customizing OGRO: The Grid Validation Policy

OGRO is configured through a set of rules -written in XML- called the Grid Validation Policy or GVP, which customizes relying parties' validation behavior. Figure 24 shows the DTD of this policy document.

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!ELEMENT ocsppolicy ( issuerdn+ ) >
3 <!ELEMENT issuerdn ( source?,unknownstatus?,
   errorhandler?,request?,proxycert? ) >
4 <!ATTLIST issuerdn dn CDATA #REQUIRED>
5 <!ATTLIST issuerdn name CDATA #REQUIRED >
6 <!ATTLIST issuerdn hash CDATA #REQUIRED >
7 <!ELEMENT revsources ( source+ ) >
8 <!ELEMENT source EMPTY >
9 <!ATTLIST source order CDATA #REQUIRED >
10 <!ATTLIST source signingcert CDATA #IMPLIED >
11 <!ATTLIST source location CDATA #REQUIRED >
12 <!ATTLIST source  type (trusted|authorized) "trusted" >
13 <!ATTLIST source timeout CDATA #IMPLIED >
14 <!ELEMENT unknownstatus EMPTY >
15 <!ATTLIST unknownstatus action (good|revoked) "revoked">
16 <!ELEMENT errorhandler ( action+ ) >
17 <!ELEMENT action EMPTY >
18 <!ATTLIST action order CDATA #REQUIRED >
19 <!ATTLIST action type
    (tryLater|setFinalResp)"setFinalResp">
20 <!ATTLIST action value (good|revoked) "revoked" >
21 <!ATTLIST action maxRetries CDATA #IMPLIED >
22 <!ELEMENT request ( signreq?, usenonce?, prot?, ext* ) >
23 <!ELEMENT signreq EMPTY >
24 <!ATTLIST signreq value (true|false) "false" >
25 <!ELEMENT usenonce EMPTY >
26 <!ATTLIST usenonce value (true|false) "true" >
27 <!ELEMENT prot EMPTY >
28 <!ATTLIST prot value (http|https) "http" >
29 <!ELEMENT ext EMPTY >
30 <!ATTLIST ext order CDATA #REQUIRED >
31 <!ATTLIST ext oid CDATA #REQUIRED >
32 <!ATTLIST ext value CDATA #REQUIRED >
33 <!ELEMENT proxycert ( unknownstatus, prevalidation ) >
34 <!ELEMENT prevalidation EMPTY>
35 <!ATTLIST prevalidation value (true|false) "false" >
36 <!ATTLIST prevalidation noprevalinfo (ocsp|ommit) "ocsp" >
```

Fig. 24. DTD of OGRO's Grid Validation Policy.

From figure 24 we can observe at line 2 that the OGRO's policy allows per issuer validation rules customization or even the option to configure a default issuer, that is, rules applying to any user whose issuer is not referenced anywhere else in the policy.

A set of revocation sources (lines 8-13) can be also defined, which means that the relying party may be able to consult more than one OCSP Responder thus providing

fault tolerance and high availability. Moreover a customizable meaning of the "Unknown" OCSP status –line 15- can be defined for any certificate on the path (Proxy or non-Proxy). Also at lines 16-19, error handling mechanisms may be declared to take a certain action if for example an OCSP Responder could not be contacted. The current set of error handlers can be extended to fulfill special requirements of VOs.
Customization of OCSP Requests (i.e. use of signatures and nonces) is provided by OGRO –lines 22 to 32- . Also important to note are Proxy Certificate's pre-validation rules (lines 34-36), which will be explained later in the next section.
More than being a set of configuration directives, OGRO's Grid Validation Policy represents a mechanism to tailor validation process' security level. For example a VO may decide to use only a defined set of internal OCSP Trusted Responders benefiting performance (i.e. not using digital signatures, nonces nor HTTPS), while other VO may use external OCSP Authorized Responders, however compelling its clients to employ "strong" OCSP Requests (i.e. digitally signed, using a nonce and over HTTPS).
At moment of writing the present thesis, OGRO was published as a "dev.globus Incubator Project" [Incubator], just in time to be included into the next major release of the Globus Toolkit. No doubt that this is a major achievement of the research reported in this document.
To demonstrate the flexibility of OGRO and its GVP, the following section compares several policies for a Grid-OCSP infrastructure based on the CertiVeR Responder.

6.2.2. Performance results

In this section are shown the performance results for a validation architecture based on the CertiVeR Service and the OGRO middleware for the Globus Toolkit 4. The setup used for the tests is described next:

- CertiVeR Validation Service:
  - Installed on a server with one Xeon processor @2.9 GHz, 1.5 Gb RAM and Windows 2000. The Responder used an Oracle database.
  - No cryptographic hardware was used in the Responder.
  - An OCSP Responder in Trusted mode for the AC CertiVeR issuer.
  - One OCSP extension is being handled: the *"CA_RATING_EXTENSION"* (registered with the OID *"1.3.6.1.4.1.4710.2.454.10.1.1"*).
  - No fault tolerant architecture is being used in the test environment.
  - Proxy certificate revocation was not configured to simulate a typical OCSP service.
  - No precomputed OCSP Responses were used.
- OGRO client:
  - Integrated into the *ProxyPathValidator* class of the Java CoG version 4, so that it could be used with the *ProxyInit* class from the same package. Remember that the same classes are used by the Globus Toolkit 4 (Java Core).
  - Apache Ant's script to run 50 Grid clients concurrently (each one under a different instance of the Java Virtual Machine) in a server with 4 Xeon processors using RedHat Linux 7.2.
  - OGRO always verifies the OCSP Response's nonce and digital signature.
  - If the validation was successful then the Proxy Certificate is written to disk.
  - The Grid Validation Policy used for all the tests described in the table 4 can be seen in figure 25.

Table 4. Tests performed with OGRO and CertiVeR into GT4

Name of the test	Explanation
No OCSP	No OCSP validation at all.
NoNonce-NoSign-HTTP	OCSP Requests without nonce, not being signed and using HTTP.
Nonce-NoSign-HTTP	OCSP Requests with nonce, not being signed and using HTTP.
Nonce-Sign-HTTP	OCSP Requests with nonce, digitally signed and using HTTP.
NoNonce-NoSign-HTTPS	OCSP Requests without nonce, not being signed and using HTTPS.
Nonce-Signed-HTTPS	OCSP Requests with nonce, digitally signed and using HTTPS.

Before the tests, our research was expecting to identify a policy with the best performance (presumably the *NoNonce-NoSign-HTTP*) and also the most secure policy (in theory the *Nonce-Signed-HTTPS*).

```
<?xml version="1.0"?>
<!DOCTYPE ocsppolicy SYSTEM "ocsppolicy.dtd">
<ocsppolicy>
  <issuerdn dn="C=ES,O=CertiVeR,2.5.4.45=2003,CN=AC CertiVeR"
            name="AC CertiVeR" hash="o6MjoB5y4b2cNvILPcBxWafHs7k=" >
    <unknownstatus action="revoked" />
    <errorhandler>
      <action order="1" type="tryLater" maxRetries="2" />
      <action order="2" type="setFinalResponse" value="revoked" />
    </errorhandler>
    <request>
      <signrequest value="true" />
      <usenonce value="true" />
      <protocol value="https" />
     <extension order="1" value="CA_RATING_EXTENSION"
        oid=" 1.3.6.1.4.1.4710.2.454.10.1.1 " />
    </request>
  </issuerdn>
  <issuerdn name="default" dn="*" hash="na" >
    <revsources>
      <source order="1" location="http://tacar.certiver.com"
        timeout="3600" type="trusted" />
      <source order="2" location="http://globus-grid.certiver.com"
        timeout="3600" type="trusted" />
    </revsources>
    <proxycert>
      <unknownstatus action="good" />
    </proxycert>
  </issuerdn>
</ocsppolicy>
```

Fig. 25. Generic Grid Validation Policy used with OGRO and CertiVeR into GT4.

From the results obtained (see figures 26 and 27) it was found that on the client-side there is really *no big difference among any of them* (in fact only a 2%-6% of variation was observed). It is also interesting to note that the use of HTTPS did not imply a visible overhead neither in OGRO nor CertiVeR. A policy commonly used by relying parties in environments like Web Services is the *Nonce-NoSign-HTTP* (which protects against replay attacks, does not identify the client to the OCSP Responder and goes over clear-text HTTP), and also gave in the tests a fair balance between security and performance.
Other important conclusions from the results shown above are the following:

- From a performance point of view, the use of nonces in the OCSP Request is almost irrelevant and it could be advisable only if CertiVeR was using precomputed Responses. Otherwise, for security reasons, it should be enabled.
- Similar to the above observation is the use of digital signatures on the OCSP Request (its use is only advisable if a special reason to justify it exists –i.e. service accountability or access control purposes-).
- On the service-side it was observed that CertiVeR kept sustained response times for all the clients. In other words, no bottlenecks were evident even though all OCSP Requests were launched in parallel.
- None of the clients required to execute the error handlers or contact the secondary revocation source as configured in the policy. This emphasizes CertiVeR's observed behavior.
- The use of OCSP validation is still time-expensive for the Grid clients, and it was observed that in some cases it took little more than twice the time to create a Proxy Certificate.

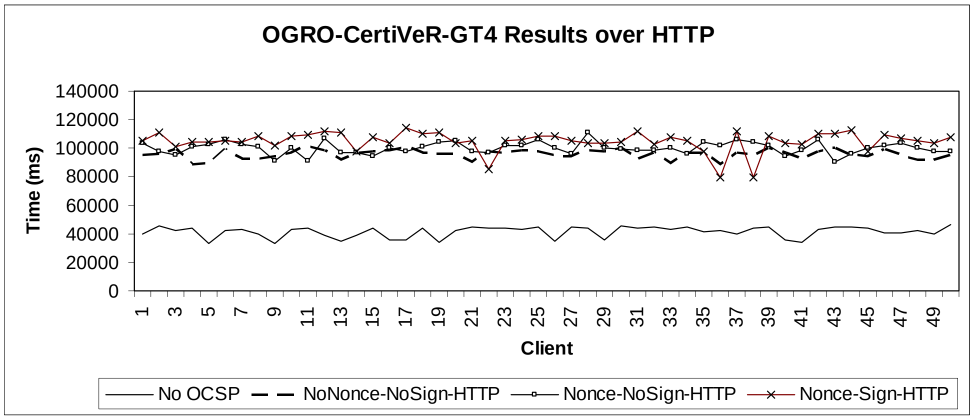


Fig. 26. Proxy Certificate initialization times with CertiVeR and different OGRO Policies over HTTP.

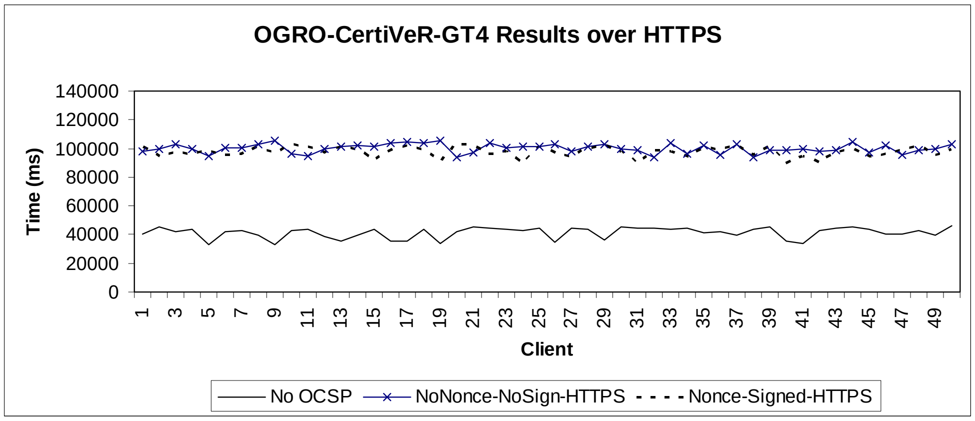


Fig. 27. Proxy Certificate initialization times with CertiVeR and different OGRO Policies over HTTPS.

Even though the whole result set showed that overall performance was pretty similar between tested configurations, the research presented in this thesis found that important improvements could be done through a couple of mechanisms: prevalidation and second-level caching, which are explained next.

6.2.3. Enhancing Grid-OCSP performance with Prevalidation

When first faced the challenge of improving Grid relying parties' OCSP performance without affecting its overall security, the authorization push model (RFC2904 [RFC2904]) was extrapolated to the OCSP arena. In RFC2904, validation information (namely "attributes" in the push model) is included along the data itself, thus creating a self-contained package able to be validated by relying parties without further contacting the source of authority (i.e. the Attribute repository). In most of the cases the relying party only requires to extract the embedded validation data, perform some cryptographic verification over it (i.e. authenticity and integrity) and finally parse it. With mechanisms like these, important performance improvements can be obtained in the validation process.

In the case of the prevalidation mechanism proposed in this thesis, what if the OCSP Response itself is embedded into the Proxy Certificate? The rationale behind this is very simple given the fact that such a message is authenticated (digitally signed by the OCSP Responder which is a trusted third party), tamper protected (again thanks to the digital signature) and includes a validity period.

OGRO and its Grid Validation Policy implement prevalidation support into both, the GT4's WSRF Container and the Commodity Grid Kit. To validate an End Entity Certificate with the prevalidation mechanism, the following procedure is performed by OGRO:

1. When creating a Proxy Certificate, the OGRO-enabled Grid client requests OCSP Validation for its whole Certificate Path to CertiVeR using the one-message mechanism (only one OCSP Request is created for the complete chain). Note that even though this OCSP Request is digitally signed, it does not require including a nonce because in any case the relying party will not verify it later.
2. As soon as the OCSP Response is received from CertiVeR, it is validated and embedded as a Proxy Certificate's X.509 extension (identified under a specific OID).
3. The OGRO-enabled Grid client invokes a Grid Service from the WSRF Container using the Secure Messaging mechanism, in such a way that an XML message -digitally signed by the prevalidated Proxy- is created.

The OGRO-enabled Grid Services WSRF Container extracts the prevalidation information from the Proxy Certificate, then performs the OCSP Response's security checks defined in RFC2560 (digital signature, trust level, etc.) and finally validates the Proxy Certificate Path according to the statuses already included into such OCSP Response.

6.2.4. Prevalidation: performance results

The results obtained with OGRO's prevalidation mechanism, under the same conditions described in section 6.2.2 are shown in Figure 28. Just as expected in the client side, the time consumed by OGRO's validation and prevalidation processes, was almost 100% above the time that elapsed when such mechanisms were not used. However it is interesting to note that embedding the OCSP Response into the Proxy Certificate did not result in a visible overhead. On the other hand, the results obtained with the Grid WSRF container –server side- (figure 29) showed that the pre-validation process reduced in little more than 30% the time required to validate a Proxy Certificate with OCSP. Even more important to note is that the pre-validation checking at the server side did not introduce a visible overhead (in fact less than 1%). On the other hand, even though a bottleneck at the WSRF Container itself was noticeable as the number of concurrent invocations increased (around the 9th invocation in figure 29), again it does not deny the fact that prevalidating improves OCSP performance and in the best of the cases (if no bottleneck was generated at the WSRF Container) the gap between the OCSP No Prevalidated and OCSP Prevalidated series (figure 29) could be reduced, but would never be the same.

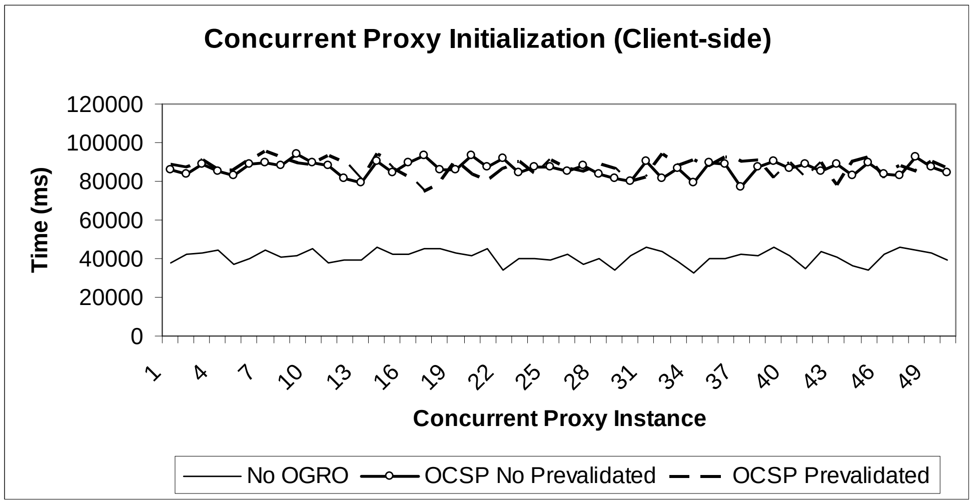


Fig. 28. Results obtained with the Grid client when creating the Proxy Certificate.

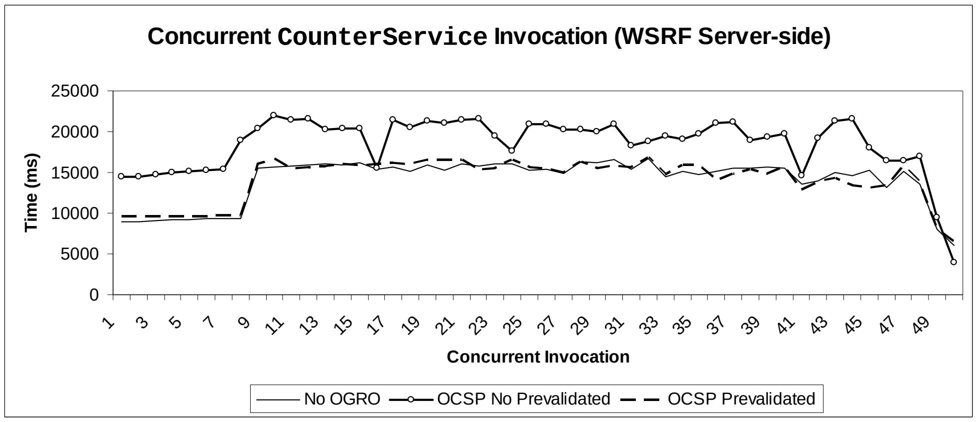


Fig. 29. Results obtained with the WSRF Container when validating the Proxy Certificate.

6.2.5. Second-level caching with OGRO

One of the recommendations given in section 3.2.3 refers to caching the OCSP Responses for enhancing validation performance. Such a cache can be extended to several levels of the Grid infrastructure to achieve this goal: a first-level cache at the Grid client, at second-level cache at the Grid server (i.e. the WSRF container) and a third-level cache at the local OCSP Server. Using such a cache hierarchy it is possible to reduce not only the cryptographic and communications overhead caused by the interaction with the global OCSP Responder, but also the total number of queries made to the server – this last aspect can also result in economic advantages considering that some OCSP providers sell their service on a per-request basis. When further considering the idea behind the OCSP cache hierarchy, we expected that second-level caching would provide the best cost-benefit and secure solution for Grids. This is due to the fact that in most Grids installations it is reasonable to think that first-level caches will serve too few clients –in the worst of the cases a cache per Grid client- when compared with a second-level cache. On the other hand, third-level caches -installed at the local OCSP Responders– may create not only bottlenecks with all the Grid servers contacting them (local Responders are not expected to be highly scalable or fault-tolerant in contraposition to a Trusted Responder serving several domains), but also might be more sensible to network failures than first- and second-level caches.

The latest version of OGRO implements OCSP validation with second-level caches through the following steps:

1. The Grid client creates its Proxy Certificate in the usual way (it does not require OGRO).
2. Cache initialization: When a Grid Service is requested from the OGRO-enabled WSRF container (also through the Secure Messaging mechanism as with prevalidation), the OCSP Responder is contacted to request validation for the whole End Entity certificate chain. As soon as the Grid server receives the OCSP Response, it is verified and stored into a local cache.
3. Cache searching: Client connections to the Grid server implying a cache-search (OGRO can be configured to enable caching only for certain Certification Authorities hierarchies) will use the native lookup methods from such data structure. One of the following will occur i) In case of a cache hit, then OGRO retrieves the stored OCSP Response, verifies it (to avoid outdated entries) and finally obtains the correspondent status for the whole certificate chain or ii) In case of a cache miss, the appropriate OCSP Responder will be contacted and the new Response stored into the cache, so subsequent calls from the same user will result in a cache hit.
4. The cache is being frequently purged by a separate thread, so stale entries (outdated OCSP Responses according to RFC 2560's verification method) are removed when detected. Responses reporting an invalid End-Entity Certificate Chain –i.e. those with at least one revoked certificate- are permanently stored.

The following two sections will provide a comparison of the results obtained with the prevalidation and the caching mechanisms, along with a security analysis of both related with the Proxy Certificate Revocation problem.

6.2.6. Performance comparison between prevalidation and caching

With the purpose of comparing OGRO's prevalidation and second-level caching mechanisms, a CertiVeR's OCSP Trusted Responder (server with one Xeon processor @2.9 GHz, 1.5 Gb RAM and Windows 2000) was set up at http://globus-grid.certiver.com. The Responder was using an Oracle database to store the revocation information, but neither cryptographic hardware nor precomputed OCSP Responses were utilized. For the Grid clients we created a set of 5 different End-Entity Certificates and set up 50 OGRO-modified Commodity Grid Kit installations with such credentials (same certificate for each 10 clients to test caching capabilities). The underlying hardware was composed of a 4-Xeon processors server running over RedHat Linux 7.2. Finally a GT4 WSRF Container was also integrated with OGRO and installed on a 2-Xeon processors server running over RedHat Linux 7.2.

Each one of the three tests reported in this section consisted of concurrently run 50 clients to request the same Grid Service from the WSRF Container through the Secure Messaging mechanism, in such a way that OCSP Validation could be triggered according to the applicable Grid Validation Policy –GVP- shown in table 5.

The results obtained from our tests are shown in figure 30, where the total GVP performance is computed as the addition of *i)* the time required by the WSRF Container to perform the OCSP call (when applicable) and *ii)* the time to cryptographically verify and parse the OCSP Response (for second-level caching it is the time to read the certificate's status from the in-memory cache, while for the prevalidation will be the time required to extract and verify the OCSP data from the Proxy certificate).

Table 5. The Grid Validation Policies being tested

Grid Validation Policy	Explanation
OCSP	WSRF Container uses OCSP to validate every Grid client request.
Cache	Second-level caching enabled at the WSRF Container. OCSP validation will be invoked only if a call to the cache is missed.
Prevalidated	Prevalidation support enabled so the WSRF Container never has to contact the OCSP Responder.

As we can see from figure 30, the use of plain OCSP -"Total OCSP" graph- conveys the highest overhead for the validation process, something already concluded in section 6.2.2. In fact the peaks observed during this test are due to bottlenecks created at the OCSP Responder when several instances from the WSRF's OGRO API are concurrently requesting validation, and afterwards when the container itself performs the cryptographic verification and parsing over each one of the received OCSP Responses. The observed performance peaks are repeated in almost regular intervals comprehending from 5 to 7 WSRF requests. This behavior is only dependent on the service and therefore, it is not a subject of study for this thesis.
On the other hand, the "Total Cache" test shows that before the 8th call it behaves almost like the "Total OCSP" graph, something that even though was expected -because the in-memory cache is still being filled- also corresponded with the performance peak observed in the first test. However once such cache contains the statuses for all the EECs, then the overall performance greatly improves as the OCSP Server is no longer required to retrieve the status from the in-memory cache. The latter operation requires a few tenths of microseconds due to the performance and concurrency-control mechanisms of the underlying OGRO API; a future enhancement may provide a more scalable database approach for implementing the OCSP cache.
Finally, the results obtained with the prevalidation mechanism (the "Total prevalidated" graph) do not involve the high peaks from the two previous tests, because it never required to contact the OCSP Server. On the other hand, we found some others peaks in the graph and, maybe more surprisingly, a slightly worse performance than with the second-level cache mechanism–once it has been filled-. This behavior is due to the fact that cached EEC's statuses are directly accessed from an in-memory data structure, while prevalidation data must be extracted from Proxy Certificates read from network sockets -thus requiring more system calls to the underlying system- while also requiring cryptographic validation.

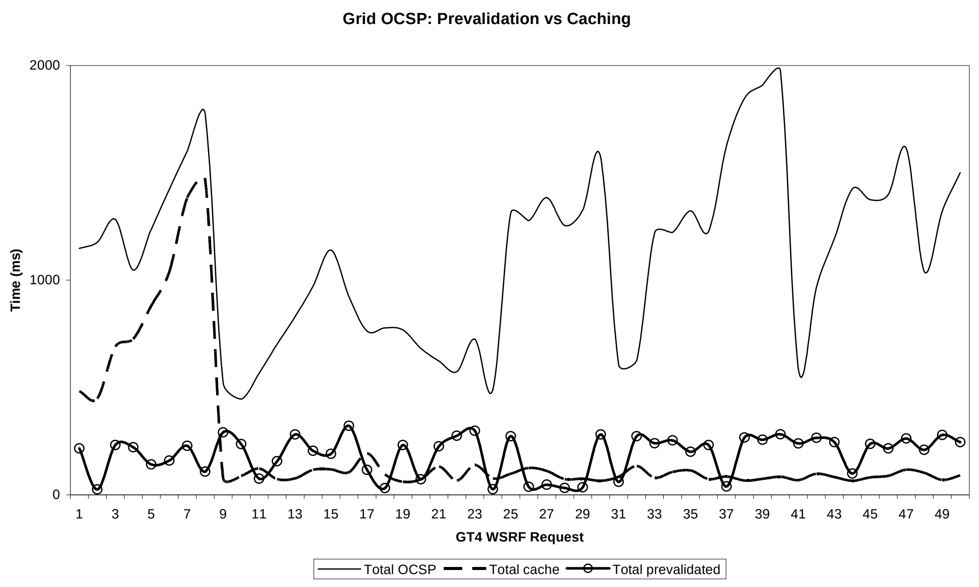


Fig. 30. Grid-OCSP performance comparison: Prevalidation versus Caching.

6.2.7. Enabling Proxy Revocation with OGRO

Proxy Certificates were intended to be short term cryptographic credentials. This is the case in most situations. However, as noted in the CAOPS-WG, there are some particular situations where Proxy certificates can have long lifetimes or they are created for future use in batch queues. Security and system administrators have discussed a number of scenarios where revoking a Proxy Certificates might be useful:

- A long-running job, that in turn creates many jobs in other domains needs to be cancelled because maybe its results are not good, it exceeds resource limits, or other problems identified by the job owner or the resource manager warrant termination. Part of the job clean-up and recovery would include revoking Proxy Certificates in use to prevent orphaned processes from running across domain boundaries.
- A system that is used as one component of a complex job is compromised. Instead of aborting the whole job and losing all the results, only the components affected by the compromised system are stopped.
- A resource owner who does not want a particular process to be run at a certain moment. Rather than blacklisting the user, assigning a low-level priority to such a process or damaging the application, the resource owner can temporally blacklist ("suspend" in OCSP language) the delegated Proxy Certificate.
- A compromised system holding a large number of proxy certificates that could have been captured by an attacker need to be revoked to avoid its malicious use anywhere in the system.

Based on the above arguments, we want to discuss in this section both approaches (the prevalidation and the second-level caches) to study how these alternatives can fulfill and even relieve the security and functional requirements of such a complex topic. This study will be developed at two different phases: the request of a Proxy Certificate status through OCSP and the revocation of a Proxy Certificate.

In first place let us suppose a scenario where the prevalidation process is performed, but when the Proxy Certificate is being created (let us call it *P*), then it is sent along with its full Certificate Path to the OCSP Responder -which is able to register *P* status- for

prevalidation. In this case we must notice that a “new” Proxy Certificate *P’* will be created back at the Grid client with such prevalidation data, but keeping the same X.509 information already present in *P* (i.e. public key, subject DN, etc.). Afterwards the Grid client only has to “register” *P’* so the OCSP Responder may be able to provide its status to interested parties afterwards. A critical security issue here is that *P’* must be registered before the embedded prevalidation data’s expiration. The notion of cautionary period (section 3.2.2) is useful in this case to avoid security issues..
Therefore it is feasible to think that when querying Proxy Certificate’s status, the Prevalidation mechanism offers a similar security level than traditional OCSP. Obviously the same holds true for the second-level cache mechanism, which will request and store the OCSP Response exactly “as is” on the Grid server. There is, however, an advantage of the prevalidation mechanism over the traditional OCSP and the second-level cache solutions: prevalidated data allows for the retrieval of a Proxy Certificate status even in situations where the OCSP Responder is not accessible from the WSRF Container domain (i.e. due to network failures).
As a second use case, if a Proxy Certificate is revoked then there will be several issues to be solved, mainly related with the entity authorized to perform such process and the means by which this revocation information is propagated through the Grid. For our purposes only the propagation problem will be considered, as OCSP by itself is not related with the mechanisms to authorize and perform the Proxy revocation. If the revocation information is not propagated to other domains in a promptly way, then there is the latent risk of keeping jobs running under the compromised Proxy control for quite a time. The prevalidation mechanism is unable to alleviate this problem, because the prevalidated data is embedded into the revoked Proxy itself. As for the second-level cache, a pretty similar problem may arise if and only if it is not being refreshed or purged constantly. Stale cache entries will represent always a security risk. The “traditional OCSP” approach answers the last Proxy Certificate status known by the OCSP Responder, however there is no guarantee that such status is the very last, as there is obviously a time required to propagate any update through the whole OCSP infrastructure. In this case, the concept of “Cautionary Period” (section 3.2.2) should be used to provide greater assurance to the OCSP Response.

6.3 POIS: Policy and OCSP based Interoperability System

The proposed POIS is comprised of three basic elements: a revocation information database, a Policy Evaluator implementing the REM technique and a high-level OCSP Responder authoritative for multiple PKI hierarchies.
At a coarse view POIS offers the following features:

1. Manage (retrieve, update) the list of CAs accredited by PMA.
2. Manage (retrieve, update) the CAs policies.
3. Manage (retrieve, update) the revocation sources and/or data.
4. Communicate with relying parties through Grid-OCSP.
5. Perform Extended Path Validation:
    - Perform Basic Path Validation.
    - Evaluate and/or Compare Policies.

In order to manage the list of accredited CAs and their policies (features 1 and 2) POIS assume an off-line communication with both PMA and CAs (the administrator manually downloads the list of accredited CAs and their policies). The revocation information (feature 3) from each accredited CA is managed using the high-level OCSP Responder like CertiVeR (see section 2.2.3).

POIS uses for its Policy Evaluator subsystem the REM API developed as part of our research (section 4.5), which allows offline evaluation of a member CA's Certification Policy to obtain its respective Global Security Level.
Using the high-level OCSP Responder's database, the obtained GSL data is linked to the existing Certificate Authority information (i.e. its revocation data). When the end-entity uses OGRO's prevalidation mechanism (section 6.2.3), the OCSP Responder embeds the Trust Anchor's GSL into the OCSP Response itself. If required, it is possible for our prototype to embed per-provision LSLs for a finer-grained comparison at the relying party. Afterwards to convey this GSL to the relying-party (feature 4), the OGRO middleware simply needs to extract the prevalidation data from the proxy certificate itself.
In our prototype, POIS is able to perform both basic and Extended Path Validation (feature 5) thanks to the OGRO middleware. Basic path validation is done using native GT4 algorithms, while the Extended Path Validation is performed by *i)* extracting certificates' statuses from the OCSP prevalidation data and *ii)* comparing the embedded GSLs according to OGRO's Grid Validation Policy explained in section 6.2.1.
Figure 31 shows a typical POIS' use case, where interoperability is achieved between a Grid Client belonging to *Institution A* and a Grid Services Container from *Institution B*. The next two sections will explain in greater detail this process from the point of view of both entities, the Grid Client and the Grid Services Container.

6.3.1. Use case 1: POIS and the End-Entity

When an End-Entity uses POIS to dynamically build Grid interoperability among different CAs, the phases depicted in figure 32 take place. If compared versus OGRO's prevalidation (figures 21 and 22), it is easy to note the addition of the GSL value in the following steps:

1. In an offline manner the CA submits its Certification Policy (CP) to the POIS, and then the Policy Evaluator subsystem feeds it to its REM API to obtain a GSL. As mentioned in the previous section, this GSL is stored into the high-level OCSP Responder's database along with the corresponding CA data.
2. The End-Entity builds an OCSP Request with a specific extension field (fully compliant with [RFC2560]), that the OCSP Responder will understand as a requirement to include its corresponding GSL –let us call it GSL_{EE}- along with the OCSP Response. Note that each CA from the End-Entity's certificate chain may be associated with a different GSL, for the REM technique this is quite feasible.
3. Finally, when the End-Entity receives the OCSP Response (with GSL_{EE} embedded also as an extension field), the prevalidation mechanism is executed to create a proxy certificate with this data embedded.

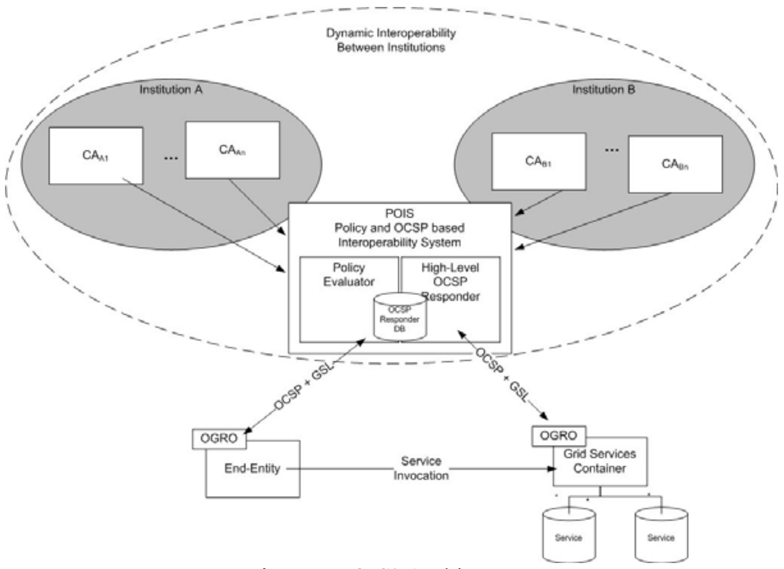


Fig. 31. POIS' Architecture.

Thanks to OGRO's prevalidation mechanism not only the relying party is able to improve the validation process' performance, but also the Proxy Certificate is self-contained in the sense that includes all the data required by relying parties to perform the extended validation process without further contacting any authority. This validation data has been digitally signed by the OCSP Responder.

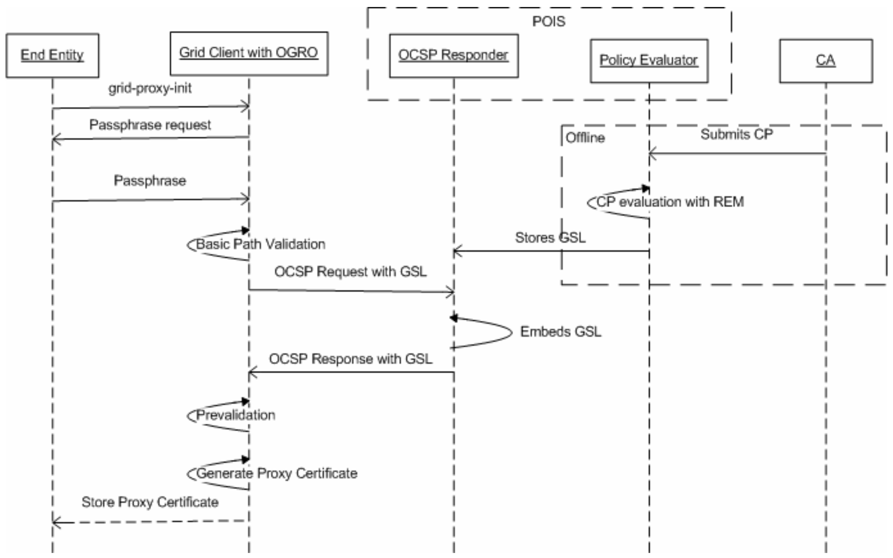


Fig. 32. End-Entity performing Extended Path Validation with POIS.

6.3.2. Use case 2: POIS and the Grid Services Container

Once the proxy certificate has been created with prevalidation data and the GSL value(s) according to the previous section, it is possible for the Grid Server to perform the interoperability evaluation in order to take a final validation and interoperability decision on the End-Entity invoking the Service's operation. Figure 33 extends the validation process previously introduced in figure 22 with the following enhancements:

1. If OGRO's prevalidation mechanism was used, then the End-Entity's GSL_{EE} is extracted from this data, Otherwise, this GSL value is requested directly from the OCSP Server.

2. The interoperability test is performed by comparing GSL_{EE} with the minimum required-GSL defined by the relying-party in OGRO's Grid Validation Policy –let us call it GSL_{SVR} -. If $GSL_{EE} \geq GSL_{SVR}$ then both Grid-CAs may interoperate.

Notice that with POIS it is possible to dynamically test for an interoperability condition on the server-side, but the End-Entity could be also able to request a minimum expected GSL from a Grid-node. This "mutual-interoperability" feature will be discussed in the next chapter, when reviewing our future work.

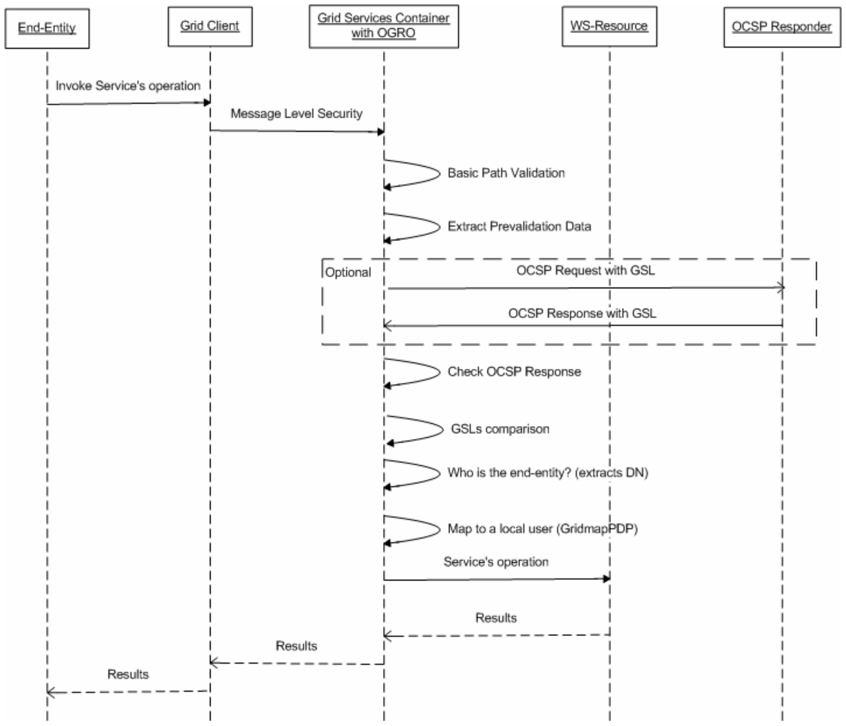


Fig. 33. Modified Server for Extended Path Validation with POIS.

6.3.3. Use case 3: a SOAP based POIS

At the moment of writing the present thesis, the use of XML protocols (i.e. SOAP) for the computational Grid was quite generalized since the birth of concepts like OGSA [OGSA]. Nobody questions anymore the flexibility of XML for implementing complex protocols conveying a wide variety of data.

Besides enabling POIS to convey its responses via OCSP (sections 6.3.1 and 6.3.2), we also designed a SOAP interface for the system, so it could be possible to hide all the OGRO management (i.e. definition of the Grid Validation Policy) to the relying parties.

Another clear advantage of this approach was the flexibility that could be achieved via this mechanism, as it may be possible to transport in a near future more validation information (besides the GSL and the OCSP Response).

The WSDL definition of POIS' Web Service interface includes an operation that can be invoked by the relying parties (the End-Entity or the Grid Services' WSRF Container) to perform an interoperability test on an end-entity certificate. In this case, the following steps would take place (see also figure 34):

1. The WSRF Container sends to POIS a SOAP message containing its own digital certificate and the one of the End-Entity requesting a Grid Service.

2. The POIS receives the SOAP message, extracts the end-entity and the WSRF's certificates and, based on their issuing CA, retrieves the GSLs associated with them. Remember that Certificate Policy evaluation is performed offline.
3. Because POIS integrates an OGRO client, checks with Grid-OCSP the statuses of both certificates (the End-Entity's and the WSRF's).
4. The SOAP response from POIS includes the following information related with the End-Entity's and the WSRF's certificates: the *final* OCSP statuses (obtained after applying OGRO's Grid Validation Policy) and the GSLs.
5. The WSRF receives the POIS' response, parses the contained information and performs the Extended Path Validation process to decide if it is possible to interoperate with the Grid client that requested the service.

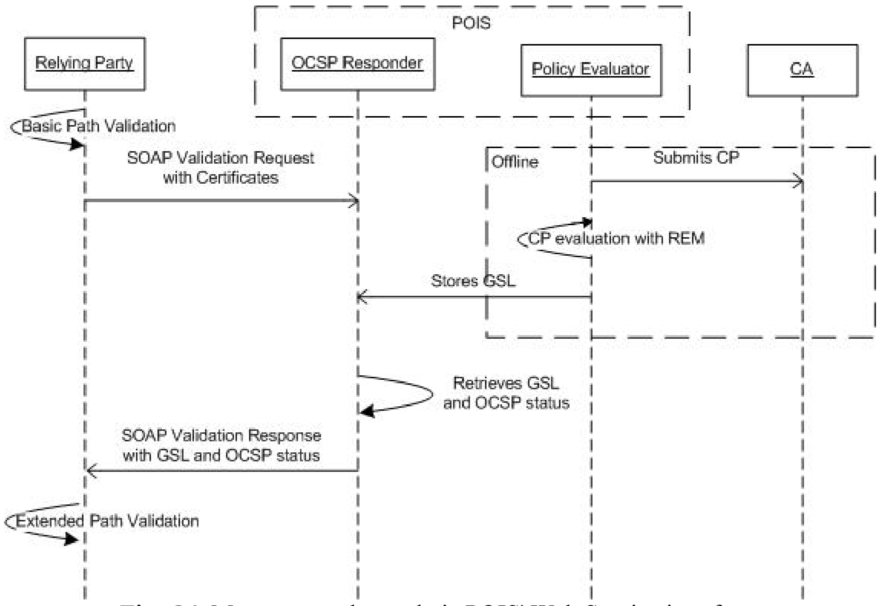


Fig. 34. Messages exchanged via POIS' Web Service interface.

Notice that in this case the requesting End-Entity does not need to contact POIS (obviously if mutual validation is not being used) and despite the WSRF does not implement an OGRO client, still it has total control over the Extended Path Validation process (i.e. it can decide under which OCSP statuses/GSLs should End-Entity access be allowed). This design feature enforces the basic idea behind a Grid authorization process: let the resource owner take the ultimate decision about who can access his resources.

6.4 Open issues

A first open-source prototype implementing a basic SOAP-based POIS has been developed as a proof of concept, using the building blocks from our previous research: the Grid-OCSP infrastructure (including the OGRO API) and the REM API. Our belief is that greater flexibility may be achieved in the future thanks to this design decision and of course, the Grid community's feedback.

However there are still important tasks that need to be solved in order to achieve a sound validation solution with POIS, in particular referring to an automatic way to audit the Certification Policy that is being evaluated with the REM API. This would provide not only greater guarantees to the system, but also may leverage the PMA's workload.

A second requirement for a production-level POIS refers to the security around the Certification Policies being retrieved. An initial idea consists in expressing the CP as an XML document, built and digitally-signed by the corresponding Certificate Authority.
The following section covers in more detail the future work related with the open issues mentioned above, and also introduces new application fields that we are beginning to explore for applying the basic ideas behind this thesis. In particular will introduce our current work for evaluating the security level of storage elements used into a Data Grid system.

CHAPTER 7 CONCLUSIONS AND FUTURE WORK

The Computational Grid relies on the assumption of sharing resources from different institutions to solve computationally intensive problems, that otherwise could need expensive and dedicated hardware systems.
Nowadays production-level Grids can be seen not only in traditional academic and research fields, but also in novel applications ranging from financial to eHealth environments. However the adoption of this paradigm also have conveyed a broad range of security concerns related with the authentication and authorization of users willing to access resources and services available through the so-called Virtual Organizations. Unfortunately, as seen at the beginning of this thesis (table 1) currently used Authentication and Authorization Infrastructures (AAIs), that have proven reliable in other environments, now face complex challenges when being deployed into the Grid. In most of the cases the main problem is related with one basic concept: *interoperability*.
In this thesis we have researched interoperability issues related with Grid authentication and authorization mechanisms, and the resulting proposal is a system able to "bridge" both concepts via an *Extended Path Validation process*. This novel concept has contributed the Grid community with the following:

- A near-real time validation of the end-entity digital certificate via a Grid-OCSP infrastructure and,
- A quantitative evaluation of the security level associated with the Certification Policy of the issuing Certificate Authority (using an extension of the Reference Evaluation Methodology –REM-).

Of course our contribution does not exclude current technologies; by the contrary, the information obtained via the proposed system is able to *complement* current AAIs to provide Grid relying parties (i.e. resource owners) with the ultimate control over access decisions related with the services and resources shared with the Virtual Organization.
Another important contribution of this thesis to the Grid community has been the implementation of an open source Grid-OCSP client (OGRO) and an API that performs the Certification Policy evaluation (REM library). Both prototypes have been used to create a Web Service that can be queried by Grid relying parties (i.e. a WSRF Grid Services' container) to take interoperability decisions (the POIS system).
At the moment of writing this thesis, we keep pushing forward the main contributions of our research into the following areas:

- Open Grid Forum (OGF): a recommendation based in our Grid-OCSP architecture is still being written, so hopefully in the following months and with more practical experience on this field, it will be possible to publish it.
- Globus Toolkit: The OGRO client developed as part of this thesis is now being tested by the Grid community as an Incubator project. So far the feedback is quite positive, so our belief is that quite soon it may be possible to have a production-level version; this is step is just prior to its integration into the next release of the Globus Toolkit.
- CoreGRID and "Second University of Naples": mainstream adoption of the "Grid-REM" technique is taking place, with a particular focus on the Data Grid just as explained at the end of this section.
- Level of Assurance: this international project [LoA] aims to develop mechanisms for evaluating the security level associated with authentication mechanisms in several environments (not only Grids). We have just begun contributing a first deliverable thanks to our experience with the "Grid-REM" technique.

As future work we are still developing a production-level POIS by introducing new security mechanisms (i.e. protection of the Certification Policies to evaluate), enhancements (i.e. convey POIS information through a SAML protocol, thus achieving compatibility with OGSA authorization profiles) and assurance mechanisms (i.e. automatic audit of most elements from the policy being evaluated).

We are also exploring new application fields that may contribute to enhance the security issues in current Grid installations, in particular those related with the Data Grid where thanks to an on-going fellowship with the CoreGRID Network of Excellent [CoreGRID] it has been possible to propose a novel mechanism to protect the data at-rest from untrusted storage elements ([SC07] and [PCGrid08]). Basically we are using the Grid-REM technique to evaluate the security level associated with the Virtual Organization's storage elements, afterwards this numeric value is used to service a minimum "Quality of Security" requested by the Grid user (in analogy with the Quality of Service requested when using a communication channel).

In general, despite this is a slow process our belief is that the Grid community is accepting quite well the main ideas behind this thesis, just as shown by the publications related with our research (see Appendix B).

References

[AACE-B1] "TERENA TF-AACE: Deliverable B.1". López, Diego. et. al. Version 1. July 2003.

[AkCom] "A comparison of the Akenti and PERMIS authorization infrastructures". Chadwick, David. et. al. 2003.

[Akenti] "Certificate-based Authorization Policy in a PKI Environment". Thompson, Mary. et. al. 2001.

[AkPol] "Akenti Policy Language". Thompson, Mary. et. al. July, 2001.

[APGridPMA] "Asia-Pacific Grid Policy Management Authority". October, 2006. http://www.apgridpma.org/

[Are05] "Towards Web Services Profiles for Trust and Security in Virtual Organizations". Arenas A., et. al. In IFIP Working Conference on Virtual Enterprises (PRO-VE'05), Valencia, Spain. 2005.

[AuthZFwk] "Conceptual Grid Authorization Framework and Classification". Lorch, Markus. et. al. Global Grid Forum. Rev. November, 2004.

[CapSys] "Capability-Based Computer Systems". Levy, Henry. Ed. Digital Press. 1984.

[Cardea] "Cardea: Dynamic Access Control in Distributed Systems". NASA Advanced Supercomputing Division. November, 2003.

[CAS] "A Community Authorization Service for Group Collaboration". Pearlman L., et. al. In Proceedings of the 4th International Workshop on Grid Computing. November, 2003.

[CernCP] "CERN Certificate Authority: Certification Policy". August, 2006. http://service-grid-ca.web.cern.ch/service-grid-ca/cp_cps/cp_cps.html

[CertiVeR] "CertiVeR: Certificate Revocation and Validation Service". November, 2006. http://www.certiver.com

[CGTrust] "D.IA.16 Update of the Survey Material on Trust and Security". CoreGRID FP6. August, 2007.

[CoG] "A Java Commodity Grid Kit". Gregor von Laszewski, Ian Foster, Jarek Gawor, and Peter Lane, Concurrency and Computation: Practice and Experience, vol. 13, no. 8-9, pp. 643-662, 2001, http:/www.cogkit.org/

[CoreGRID] "CoreGRID Network of Excellence". November 2007, http://www.conregrid.net

[DoECP] "US Department of energy Grids: Certification Policy". August, 2006. http://www.doegrids.org/Docs/CP-CPS.pdf

[EUGridPMA] "European Policy Management Authority for Grid Authentication". Octuber, 2006. http://www.eugridpma.org/

[Gam00a] "Trust: Making and Breaking Cooperative Relations". Gambetta D. (ed). Department of Sociology, University of Oxford, 2000. http://www.sociology.ox.ac.uk/papers/trustbook.html .

[Geu05] "Web Services and Web Service Security Standards". Geuer-Pollmann C. and Claessens J. Information Security Technical Report, Elsevier, 10(1)15:24, 2005.

[GFD16] "Global Grid Forum Certification Policy Model". Butler, R., Genovese, T. Global Grid Forum. June, 2003. http://www.ggf.org/documents/GFD.16.pdf

[Globus]	"Globus Toolkit Version 4: Software for Service-Oriented Systems". I. Foster. IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13, 2005.
[Gra00]	"A Survey of Trust in Internet Applications". Grandison T. and Sloman M. In IEEE Communications Survey and Tutorials, 3, 2000.
[GridAP]	"Classic AP Profile Version 4.03". Approved by the EUGridPMA. Edited by David Groep. 2005. http://wwww.eugridpma.org/igtf/IGTF-AP-classic-20050905-4-03.pdf
[GT4SEC]	"GT 4.0: Security: Message & Transport Level Security". 2006. http://www.globus.org/toolkit/docs/4.0/security/message/
[IGTF]	"International Grid Trust Federation". October, 2006. http://www.gridpma.org/
[Incubator]	"dev.globus Incubator" November, 2006. http://dev.globus.org/wiki/Welcome
[InfnCP]	"INFN Certificate Authority: Certification Policy". August, 2006. http://security.fi.infn.it/CA/CPS/
[IrisCP]	"pkIRIS Grid Certificate Authority: Certification Policy". August, 2006. http://www.irisgrid.es/pki/policy/
[Jos07]	"A Survey of Trust and Reputation Systems for Online Service Provision". Josang A., Ismail R. and Boyd C. In Decision Support Systems, 43(2), pp 618-644, 2007.
[Lin04]	"Liberty Trust Models Guidelines". Linn J. (ed). Liberty Alliance Project, version 1.0, 2004.
[LoA]	"ES-LoA Project Output". October, 2007. http://www.es-loa.org/
[OCSPReq]	"OCSP Requirements for Grids". Luna, J. et. al. Open Grid Forum, CA Operations Work Group. Working Document. May, 2005. https://forge.gridforum.org/projects/caops-wg
[OGF]	"The Open Grid Forum". November, 2006. http://www.ogf.org/
[OGRO]	"The Open GRid Ocsp API". November, 2006. http://globus-grid.certiver.com/info/ogro/
[OGSA]	"The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration". Foster, I., Kesselman, C., Nick, J. and Tuecke, S. Globus Project, 2002, http://www.globus.org/research/papers/ogsa.pdf
[OGSASEC]	"The Security Architecture for Open Grid Services". Nagaratnam N., et. al. 2002. http://www.cs.virginia.edu/~humphrey/ogsa-sec-wg/OGSA-SecArch-v1-07192002.pdf
[OSSTMM]	"OSSTMM: Open Source Security Testing Methodology Manual." August, 2006. http://www.isecom.info/mirror/osstmm.en.2.1.1.pdf
[PCGrid08]	"Providing security to the Desktop Data Grid". Luna J., et. al. Submitted to the CoreGRID PCGrid 2008 Workshop. November, 2007.
[Permis]	"The PERMIS X.509 Role based privilege management infrastructure". Chadwick, David. Otenko, Alexander. June, 2002.
[PRIMA]	"The PRIMA Grid Authorization System". Lorch M. and Kafura D. In Journal of Grid Computing. Springer Netherlands. Vol. 2, no. 3. pp- 279-298. September, 2004.
[REM1]	"An innovative Policy-based Cross Certification methodology for Public Key Infrastructures". Casola V., Mazzeo A., Mazzocca N., Rak M. In 2nd EuroPKI Workshop. Springer-Verlag LNCS 3545, pp 100-

117, Editors: David Chadwick, Gansen Zhao. 2005. http://sec.cs.kent.ac.uk/europki2005/

[REM2] "A Reference Model for Security Level Evaluation: Policy and Fuzzy Techniques". Casola V., Preziosi R., Rak M., Troiano L. JUCS - Journal of Universal Computer Science. Editors: Ajith Abraham, L.C. January, 2005.

[REM3] "A SLA evaluation methodology in Service Oriented Architectures". Casola V., Mazzeo A., Mazzocca N., Rak M" in Proceedings of Quality of Protection Workshop 05, 15 September, 2005, Milan, in Advances in Information Security book series, Springer-Verlag

[REM4] "A Policy Based Methodology for the Analysis, Modelling and Implementation of Security Infrastructures". Casola V., PhD Thesis, Second University of Naples, 2004.

[RFC2078] "RFC 2078: Generic security service application program interface – GSS API version 2". Linn J. Internet Engineering Task Force. 1997. http://www.ietf.org/rfc/rfc2078.txt

[RFC2246] "RFC 2246: The TLS Protocol version 1.0". Dierks T., and Allen C. Internet Engineering Task Force. 1999. http://www.ietf.org/rfc/rfc2246.txt

[RFC2560] "RFC 2560: X.509 Internet Public Key Infrastructure, Online Certificate Status Protocol – OCSP". Myers M, et. al. June 1999. http://www.ietf.org/rfc/rfc2560.txt

[RFC2904] "RFC 2904: AAA Authorization Framework". Vollbrecht J, et. al. August Internet Engineering Task Force. 2000. http://www.ietf.org/rfc/rfc2094.txt

[RFC3280] "RFC 3280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile". Housley R., et. al. Internet Engineering Task Force. 2002. http://www.ietf.org/rfc/rfc3280.txt

[RFC3281] "RFC 3281: An Internet Attribute Certificate Profile for Authorization". Farrel S. and Housley R. Internet Engineering Task Force. 2002. http://www.ietf.org/rfc/rfc3281.txt

[RFC3379] "RFC 3379: Delegated Path Validation and Delegated Path Discover. Protocol Requirements". Pinkas D. and Housley R. Internet Engineering Task Force. 2002. http://www.ietf.org/rfc/rfc3379.txt

[RFC3647] "RFC 3647: Internet X.509 Public Key Infrastructure, Certification Policy and Certification Practices Framework". Chokhani S., et. al. November 2003. http://www.ietf.org/rfc/rfc3647.txt?number=3647

[RFC3820] "Internet X.509 Public Key Infrastructure proxy certificate profile". Tuecke S., et. al. Internet Engineering Task Force. 2004. http://www.ietf.org/rfc/rfc3820.txt

[SAML] "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V1.1". OASIS Security Services Technical Committee. Version 1.1. September, 2003.

[SC07] "An analysis of security services in grid storage systems". Luna J., et. al. In CoreGRID Workshop on Grid Middleware 2007, June 2007.

[SCVP] "Server-based Certificate Validation Protocol". Housley R., et. al. Internet Engineering Task Force Draft. September, 2007. http://www.ietf.org/internet-drafts/draft-ietf-pkix-scvp-33.txt

[Shib] "Shibboleth Overview and requirements". Carmody, Steven. February, 2001.

[TAGPMA]	"The Americas Grid Policy Management Authority". October, 2006. http://www.tagpma.org/
[UniAAI]	"Towards a Unified Authentication and Authorization Infrastructure for Grid Services: Implementing an enhanced OCSP Service Provider into GT4". Luna J., Manso O., Manel M. 2nd EuroPKI 2005 Workshop. Proceedings by Springer in Lecture Notes in Computer Science series. July 2005. http://sec.cs.kent.ac.uk/europki2005/
[VOMS]	"VOMS Architecture v1.1". EU DataGrid. March, 2007. http://grid-auth.infn.it/docs/VOMS-v1_1.pdf
[WSRF]	"The WS-Resource Framework". 2006. http://www.globus.org/wsrf/
[X509]	"ITU-T X.509 (03/00): Information technology – Open systems interconnection, the directory: Authentication framework". ITU, Recommendation. 2002.
[XACML]	"OASIS eXtensible Access Control Markup Language (XACML) 2.0". Moses, Tim. et. al. December, 2004.
[XMLDSig]	"XML-Signature Syntax and Processing". Bartel, Mark. et. al. W3C Recommendation. February, 2002. http://www.w3.org/TR/xmldsig-core/

Appendix A: Global Grid Forum’s CP Model’s formalized template

Section	Subsection		Provision	Cardinality	Possible Values
Introduction			Identification	2	noODI<OIDassigned
	Community & applicability		Certificate Authority	2	CAhasRArole<independentRA
			Local Registration Authorities (LRAs)	3	noRA<onlineRA<physicalRA
			End Entities	1	CPspecified
			Policy applicability	4	AuthRelationship<AuthBusinessTransaction< AuthBusinessFinancialTransaction<AuthLegalBusFinTransaction
			Notification of certificate issuance and revocation	4	CAavailable< CA-CRLavailableOncontract < Revocation Notified < CA-CRL available
	Contact details		Specification Administration Organization	1	CPspecified
			Contact persons	1	CPspecified
			Person determining CPS suitability for the Policy	2	LocalPMA<PMA
General provisions	Obligations	CA obligations	Time between certificate request and issuance	4	5 days < 2 days < 1 day < 4 hours
			Certificate revocation and renewal	1	CAObligedToRespectPolicy
			Notification upon private key compromise	1	SubscriberObligedTo
			Revocation check responsibility	1	ObligedTOcheck
			Limitations of liability	4	noLiability<5000<50000<1000000$
			RA obligations	2	AuthenticateRequestor<ProofSubjectPubKeyPossesion
			Subscriber obligations	2	SecureCreationPrivateKey<ProtectPrivateKey
			Relying party obligations	2	CheckCertPurpose<CheckCertValidity
			Repository Obligations	2	BestEffortRepository<HighAvailabilityRepository
	Liability		CA liability	3	noLiability<minimumCAPrivateKeyProtection<CAfailureToComplyCP
			RA liability	2	noLiability<verifyIdentityOfEveryRequestor
	Financial Responsability		Indemnification by relying parties	1	definedByCP
			Fiduciary relationships	1	definedByCP
			Administrative Processes	1	definedByCP
	Interpretation and		Governing Law	1	definedByCP

	Enforcement			
		Severability,survival,merger,notice	3	notifyCACeaseOfOperation<notifyCrossCertifiedCA< revokeIssuedCertificatesAtTermination
		Dispute resolution procedures	2	definedByCP<PMADefined
	Fees	Certificate issuance or renewal fee	1	definedByCP
		Certificate access fees	1	definedByCP
		Revocation or status information access fees	1	definedByCP
		Fees for other services	1	definedByCP
		Refund Policy	1	definedByCP
	Publication and Repository	Publication of CA information	2	offLineRepository<secureOnlineRepository
		Frequency of publication	2	CertsPublishedAsIssued<CPSpublishedAsApproved
		Access Control	2	noAccessControl<AccessControlToIssuedCerts
		Repositories	2	noRepositories<onlineRepositories
	Compliance audit	Frequency of Entity Compliance Audit	2	beforeInitialAproval<OncePerYear
		Identity and qualifications of auditor	2	relatedWithAuditedCA<NoRelatedWithAuditedCA
		Auditors relationship with audited party	1	definedInCP
		Topic covered by audit	1	ComplianceCP/CPS
		Deficiency	2	relyingPartyRefusesCA<CAcannotBePartOfHierarchy
		Communication of results	3	definedInCP<resultPublicilyAvailable<publishedPMA
	Confidentiality	Confidential Information	1	EnsurePrivacyProtection(Law)
		Information considered confidential	2	NotIncludedInCertificate<ExplicitlyDeclaredConfidential
		Certificate Revocation or suspension information	2	notConsideredConfidential<EnsureConfidentiality
		Release to Law Enforcement Officials	1	DiscloseNationalLaws
		Release as Part of Civil Discovery	1	DisclosetoCivilDiscovery
		Disclosure upon Owner's Request	1	DiscloseAfterOwnerAuthorization
		Other Information Release Circumstances	1	SpecialEventsForDisclose
		Intellectual Property Rights	1	NoCLaimOnIssuedCertificates
Identification and authentication	Initial registration	Types of names	1	RFC2459Conformant
		Need for names to be meaningful	2	AssociatedUnicID<Associated

		Interpretation of name forms	1	Subject-IssuerObliged
		Uniqueness of Names	1	uniqueDNs
		Name claim dispute	1	definedByCP
		Trademarks	1	definedByCP
		Proof of possession of Private Key	2	CAgeneratesPrivKey<secureProofMethod
		Authentication of organization identity	4	OnLineAuth < InternalMechanisms < IDCard-delegation < Face-to-Face
		Authentication of individual identity	4	OnLineAuth < InternalMechanisms < IDCard-delegation < Face-to-Face
		Routine Rekey	4	identityEstablishedwithCurrentKey< identityReestablishedOnceEvery15y< identityReestablishedOnceEvery10y<i dentityReestablishedInPerson
		Rekey after revocation	4	notAllowed<UseDigSigRequest<RekeyAuthn< UseInitialRegistrationMechanisms
		Revocation Request	2	UseDigSigRequest<UseInitialRegistrationMechanisms
Operation Requirements		Certificate application	2	CAGeneratesKeyPair<EntityGeneratesKeyPair
		Certificate Issuance	1	CAnotifyRequester
		Certificate Acceptance	1	CPspecified
	Certificate suspension and revocation	Circumstances for revocation	5	noRequested<InformationChanges< SuspectCompromisePrivKey<SuspectCompromiseKeyStorage <CPChanges
		Who can request revocation	4	Anyone<theCA<theSubject<LRAtrustedRole
		Procedure for revocation request	4	byFAX<byPHONEorE-maill<byAuthenticatedProcedure< Face-to-FaceRequest
		Revocation request grace period	4	24hours<12hours<4hours<immediately
		Circunstances for Suspension	2	notSupported<SupportedByCA
		Request for Suspension	1	AcceptsRequests
		Procedure for Suspension Request	2	DigSigRequest<Face2Face
		Limits on Suspension Period	1	CPspecified
		CRL issuance frequency	4	24hours<12hours<4hours<immediately
		CRL checking requirements	1	RelyingPartiesAreObligedToCheck
		Online Revocation and Status Checking	3	notSupported<onlineCRL<OCSP
		Online Revocation Checking Requirements	1	alwaysCheck
		Other forms of Revocation	1	CPspecified

		Advertisements		
		Checking Requirements for Other Forms…	1	CPspecified
	Security Audit Procedures	Types of Event Recorded	1	AllEvents-OperationsRecorded
		Frequency of processing log	4	1 month<2weeks<1week<daily
		Retention Period for Audit Log	1	3 years
		Protection of Audit Log	1	BestEffort
		Audit Log Backup Procedures	1	CPspecified
		Audit Collection System	2	internal<external
		Notification to Event-Causing Subject	1	CPspecified
		Vulnerability Assessments	1	CPspecified
	Records Archival	Types of events Recorded	2	IssuedCertRecord<IssuedCert-CRL
		Retention Period for Archive	1	3 years
		Protection of Archive	1	internalAccessOnly
		Archive backup procedures	1	CPspecified
		Time Stamping of Records	1	CPspecified
		Archive Collection System	2	internal<external
		Procedure to obtain/verify archive info	1	CPspecified
		Key changeover	2	AutomaticChangeOver<ProcedureToRe-issueCert
	Compromise & Disaster Recovery	CA private key compromised	4	documented<RequestCACertRevocatiion<InformSubscribers< TerminateServices
		Computing resources, software and/or data corrupted	1	CPspecified
		Entity Public Key is Revoked	1	CPspecified
		Entity Key Compromised	1	CPspecified
		Secure facility after disaster	1	documented
		CA termination	4	InformSubscribers<TerminateServices< PublishInfoAfterTermination<DestroyPrivKeys+AllCopies
Physical, procedural and personnel security	Physical controls	Site location, construction and physical access	4	MonitorAuthorizedPersonel<ControlAccesstoCAservers< ControlAccesstoCA-RA-servers<CA-HighSecZone
		Physical Access	1	OnlyAuthzPersons
		Power and Air Conditioning	1	CPspecified
		Water Exposure	1	CPspecified
		Fire prevention and Protection	1	CPspecified

		Media Storage	1	CPspecified
		Waste Disposal	1	CPspecified
		Off-Site Backup	2	NotUsed<cAccessToAuthzPersons
	Procedural controls	CA trusted roles	3	Everyone<SeparationDuties2Persons< SeparationDuties3Persons
		Number of persons req per task	2	1Person<MoreThanOnePerson
		Identification and Authn for each role	1	CPspecified
	Personnel Controls	Background,qualifications,exp and clearance	2	TrainedPerson<TrainedSeparationDuties
		Background check procedures	1	CPspecified
		Training reqs	1	CPspecified
		Retrain frequency and requirements	1	CPspecified
		Job rotation freq and sequence	1	CPspecified
		Sanctions for unauthz actions	1	CPspecified
		Contracting personnel reqs	1	CPspecified
		Docs supplied to personnel	1	CPspecified
Technical security controls		Key pair generation and installation	2	generatedByCA<generatedByEntity
		Private key delivery to Entity	3	EnclosedEnvelope<EnclodeEnvelopeSplittedBtPIN< OwnedByEEC
		Public key delivery to cert issuer	1	SecureProcedure
		CA public key delivery to users	1	OnLine
		Asymmetric key sizes	3	512<1024<2048
		Generation of Public Key Params	1	PMA-approvedAlghorithm
		Parameter Quality Checking	1	CPspecified
		Hardware/software key generation	2	SoftHardModule<HwCryptoModule
		Key usage purposes (as per X.509v3 field)	3	Authentication<DigitalSignature<DigSignLegalValidity
		Private key multiperson Control	1	SeparationDuties
		Private key escrow	1	noEscrow
		Private key backup	2	storedSecureMedium<NotAllowed
		Usage periods for the public and private keys	3	6years<2years<1year
	Private key protection	System development controls	3	VerifiableDevelopmentProcess< VerifiableQualifiedDevelopmentProcess< VerifiableQualifiedDevelopmentProcess+RiskAssessment

		Security management controls	2	DocumentedUpgrades<FormalMangementMethodology
		Cryptographic module engineering Controls	3	ValidatedCryptographicModule(FIPS-140-L1)< HardwareCryptographicModule(FIPS-140-L2)< CryptographicModule(FIPS-140-L3)
	Other aspects of key pair management	Network security Controls	1	NetworkProtectionAgainstAttack
CA Certificates	Private key protection	Standards for cryptographic modules	4	SWModule<ValidatedCryptographicModule(FIPS-140-L1)< HardwareCryptographicModule(FIPS-140-L2)<CryptographicModule(FIPS-140-L3)
		Private key multiperson Control	1	SeparationDuties
		Private Key Escrow	1	noEscrow
		Private Key Backup	2	storedSecureMedium<NotAllowed
		Private Key Archive	2	onlyEncr&DecPrivKey<SigningKey
		Private Key Entry into a Crypto Module	2	SWModule<HWModule
		Activating a Private Key	4	PIN<Passprahse<HWToken<Biometric
		Deactivating a Private Key	4	PIN<Passprahse<HWToken<Biometric
		Destroying a Private Key	2	EraseFromRAM<DestroyAllCopies
	Other aspects of CA key pair management	Public Key Archival	2	RepositoryIntegrityControls<RepositoryDigSigned
		Usage periods for the public and private keys	3	20years<6years<2years
		CA Key Access Control	3	PassPhrase<HardwareToken<BiometricSensors
	Activation Data	Activation data generation and installation	4	PIN<Passprahse<HWToken<Biometric
		Activation data protection	2	Confidentiallity<Integrity
		Other aspects of activation data	1	CPspecified
	Computer security controls	Computer security technical requirements	2	dedicated<offline
		Computer security rating	1	CPspecified
	Life-cycle technical controls	System development controls	3	VerifiableDevelopmentProcess< VerifiableQualifiedDevelopmentProcess< VerifiableQualifiedDevelopmentProcess+RiskAssessment
		Security management controls	2	DocumentedUpgrades<FormalMangementMethodology
		Life-cycle security rating	1	CPspecified
		Network security Controls	1	NetworkProtectionAgainstAttack
		Cryptographic module engineering Controls	3	ValidatedCryptographicModule(FIPS-140-L1)< HardwareCryptographicModule(FIPS-140-L2)< CryptographicModule(FIPS-140-L3)
Certificate and CRL		Certificate Profile	1	X.509v3

profiles				
	CRL Profile	Version numbers	1	FixedTo1
		CRL and CRL Entry extensions	1	CPspecified
Administration of Specifications		Specification Changes	1	NotifyCA-Users
		Publication and Notification Policies	2	PMA<IGTF
		CPS Approval Procedures	1	PMA
Others		CA-RA Communication	2	Unsecure<Secure
		Disaster Recovery Procedure	2	Documented<DiscussedPMA

Appendix B: Published Material

- Journals:
  - “Using OGRO and CertiVeR to improve OCSP validation for Grids”. Luna J., Manso O., Manel M. 1st Grid and Pervasive Conference. 2007. Springer Netherlands: Journal of Supercomputing: special issue Technology Deployments in Grid Computing. ISSN 0920-8542 and 1573-0484 on line.

- Workshops and Conferences:
  - “Towards a Unified Authentication and Authorization Infrastructure for Grid Services: Implementing an enhanced OCSP Service Provider into GT4”. Luna J., Manso O., Manel M. 2nd EuroPKI 2005 Workshop. 2005. Springer-Verlag LNCS Vol. 3545. ISBN 978-3-540-28062-0. pp. 36-54. Editors: David Chadwick, Gansen Zhao. http://sec.cs.kent.ac.uk/europki2005/
  - “Using OGRO and CertiVeR to improve OCSP validation for Grids”. Luna J., Manso O., Manel M. 1st Grid and Pervasive Conference. 2006. Springer-Verlag LNCS Vol. 3947. ISBN 3-540-33809-8. pp. 12-21. Editors: Yeh-Ching Chung, José E. Moreira. http://hpc.csie.thu.edu.tw/gpc2006/ - *Best Paper Award.*
  - “OCSP for Grids: Comparing Prevalidation versus Caching”. Luna J., Manso O., Manel M. In 7th IEEE/ACM International Conference on Grid Computing, Barcelona. 2006. ISBN: 1-4244-0344-8. pp. 202-209. http://www.grid2006.org/
  - “Static evaluation of Certificate Policies for GRID PKIs interoperability”. Luna J., Casola V., et. al. Accepted for 2nd International Conference on Availability, Reliability and Security. 2007. http://www.ares-conference.eu/
  - “Interoperable Grid PKIs among Untrusted Domains: An Architectural Proposal”. Luna J., Casola V., et. al. Submitted for 2nd Grid and Pervasive Conference. 2007. http://www-lipn.univ-paris13.fr/GPC2007/
  - “An analysis of security services in grid storage systems”. Luna J., et. al. In CoreGRID Workshop on Grid Middleware 2007, June 2007. http://www.coregrid.net
  - “Providing security to the Desktop Data Grid”. Luna J., et. al. Accepted for the 2nd Workshop on Desktop Grids and Volunteer Computing Systems (PCGrid2008). December, 2007.
  - “A data-centric security analysis of ICGrid”. Luna J., et. al. Accepted for the CoreGRID Integration Workshop 2008. January, 2008.

- Extended abstracts:
  - “Using OGRO to implement OCSP certificate validation in the Globus Toolkit 4”. Luna, Jesús. Manso, Oscar. Manel, Medina. 22nd APAN Grid Meeting. Singapore. July 2006. http://www.apan.net/meetings/singapore2006/

- Recommendations and Open Source:
    - "OCSP Requirements for Grids". Global Grid Forum, CA Operations Work Group. Work in Progress. February 2007. https://forge.gridforum.org/projects/caops-wg
    - "OGRO - The Open GRid Ocsp client API". February 2007. http://dev.globus.org