Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

The (tiered) grid and its security challenges

G. Laccetti, G. Schmid

Abstract: *This work focuses on the analysis of current grid security technologies. Its aim is to establish if, where and how those technologies fail to meet the environmental requirements for grids, indicating possible areas of future research. Grid security architectures are examined with respect to an abstract model deduced by the Globus Toolkit, but which conceives grids as tiered objects. A tiered approach naturally reflects the creation and assessment of grid environments, allowing for implementation designs which are easier and safer to manage.*

RT-ICAR-NA-05-12 Luglio 2005

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sede di Napoli, Via P. Castellino 111, 80131 Napoli,
URL: *www.na.icar.cnr.it*

1.Introduction.

A grid is a distributed computing architecture for delivering computing and data resource as a service, using Internet hardware and software infrastructure to build the communication and control middleware of distributed systems, and using a Web browser as the user interface to grid-enabled applications (gridware).

As such, grids must be considered a natural evolution of previous distributed computing environments, such as Multiprocessor Systems (MPs) and Clusters of Workstations (CoWs). Indeed, both MPs and CoWs are distributed computing architectures with the following three distinguished features: small (if any) hardware and software differences among computing nodes, interconnection through shared memory or a crossbar switch network, and administration as a single system in one single administrative domain; whereas a computational grid may consist of clusters of networked single hosts, MPs, CoWs, in multiple different administrative domains, scattered over departments, enterprises, or distributed globally even over the Internet.

In the research community the term "grid" refers usually to global grids, since those were quickly recognized as a compelling, appealing evolution both of the world-wide Web and high performance computing environments. As a matter of fact, global grids are extensively studied in many papers, and various global grid infrastructures are currently under development within the framework of relevant national and multinational projects (e.g. EGEE [EGEE], NINF [NINF], NPACI [NPACI], this last one ended in 2004). Advanced prototype global grid middleware exist, both in the form of open-source (e.g. Legion [LEGIO], Globus [GLOB], UNICORE [UNIC]) and commercial (e.g. Sybase Avaki [AVAKI]) software. However, as we are going to delineate in this introduction, the architectural approach introduced for global grids allow grids to be designed and implemented as tiered objects, which is particularly appropriate with respect to security.

Global grid architectures are indeed based on the "hourglass model" [FOS01], the same model used in the deployment of the Internet: the narrow neck of the hourglass defines a small set of core abstractions and protocols, onto which many different applications can be mapped and which themselves can be mapped onto many different technologies. Moreover, the *Globus Toolkit* [FOS99]- which nowadays is the *de facto* standard middleware for global computational grids - was designed and implemented with respect to a layered architecture as the TCP/IP protocol suite and which provides the following five layers: *Application*, *Collective*, *Resource*, *Connectivity* and *Fabric*. The Globus Toolkit constitutes a (partial) implementation of Collective, Resource and Connectivity layers, which in turn represent the neck of our hourglass. These layers define protocols for the secure negotiation, initiation, monitoring, control, accounting, and payment of sharing operations on individual resources and to capture interactions across collections of such resources. At the bottom side of the hourglass there is the grid Fabric layer, which implements the local, resource-specific operations that occur on specific resources, as a result of operations at higher levels [FOS01]. A resource may be a logical entity, such as a MP or a CoW, and its implementation usually involves internal services and protocols, but these are not the concern of the global grid architecture. A resource could also be a fully distributed computing architecture, which uses Internet hardware and software to build communications and control, and Web technologies to interface to its applications: in other words, such a resource could also be itself a *grid*, in the sense of the definition given at the beginning of this section.

Thus grids can be represented as a tiered structure, in which they are organized with respect to their dimensional scale. A tiered approach to grid architecture is useful, in a similar manner that it is useful to consider layered networks.

First of all, local grid technologies have their own motivation and application, no matter if they are employed to give rise to grids embedded in wider ones. Indeed, many experiences in these last years show that they can be fruitfully employed to improve resource utilization and integration, and nowadays a lot of both commercial and open-source software environment and toolkits exist to address those requirements (e. g., *Condor* [COND], *Digipede Network* [DIGI], *Moab Grid Suite* [MOAB], *Oracle Real Application Cluster* [ORAC], *Sun N1 Grid Engine* [SUN1]).

Second, a layered implementation naturally reflects the creation and assessment of grid environments, since grids are not constructed from scratch, but they are obtained as a coordinated

resource sharing among existing collections of individuals, institutions and resources.
Finally, but with a major relevance from our viewpoint, a tiered approach to grid design allows for implementations which are easier and safer to manage: connecting each node to a global grid on the Internet is uselessly dangerous if the same functions and services can be achieved with only few hosts belonging to a global grid and the others inter-operating via a local one.
The tiered structure of grids was pointed out in [GEN01, COO02] and is at the basis of the approach followed in the last years at Sun Microsystems to network computing and grid technologies [ARK04] . Following this approach, grid computing can be divided into the following three logical level of deployment: *Global*, *Enterprise* and *Cluster* Grids. A cluster grid is the simplest form of a grid, and provides a compute service to a group or department level. The point to focus here is that a cluster grid constitutes a single administrative domain, in the sense that there are a single Policy Authority and a unique, global accounting and access control policy for all the nodes in the cluster. Moreover, cluster grids are built upon System Area Network (SAN) and/or Local Area Network (LAN) technologies[1]. Enterprise grids enable multiple projects or departments to share resources within the same enterprise or organization and don't necessarily address the security and global policy management issues associated with global grids. With respect to that, enterprise grids are composed of (possibly many) administrative domains, but all of them are deployed and enforced within the global accounting and access control policy implemented at the enterprise level. Enterprise grids are faced with the use of the Internet, though tunneling techniques such as virtual private networks (VPN) can be enforced at this level for security and reliability. Finally, Global grids are collections of enterprise and cluster grids as well as other geographically distributed resources, all of which have agreed upon global usage policies and protocols to enable resource sharing.
It should be clear that security issues greatly vary depending on the tier of grid considered. At the top of the scale measuring security requirements are global grids, whose dynamic and multi-institutional nature introduce challenging security issues that demand new technical approaches. Scaling down, we first encounter enterprise grids and then, at the bottom of the scale, cluster grids. It could appear that these two less complex environments do not require too much attention with respect to security issues, as their security infrastructure could be derived with minor changes from the more complex one implemented at the global tier. This misleading reasoning breaks up some founding principles of computing security, first of all the *Principle of Layered Defense* and the *Principle of Weakest Link* [PFL03], whose implications could be summarize as follows: although it could be appropriate to overlap controls, in the expectation that one control will compensate for the failure of another, overlapping computer security mechanisms should not result simply in redundancy (that is, straightforward mechanism duplication), since redundancy could increase administrative tasks, lowers easy of use and not offer actual improvements against attacks, not even those based on the brute force approach. Moreover, (computational) grids are entailed to offer heavy throughput computing, distributing many compute jobs onto different resources of the grid, and composing individual results back into one global result. Thus grid security mechanisms should be designed and implemented with a constant attention to performance and interoperability issues, carefully minimizing their overhead.
The main aim of this paper is to establish if, where and how current grid security technologies fail to meet the requirements imposed for such computing environments, indicating possible alternatives and areas of future research. Our analysis is carried out with respect to an abstract model for grid security architecture which accomplishes the tiered layering previously sketched and takes into account the Globus Toolkit (GT) [GLOB] as a concrete grid implementation of reference.

2.Grid security challenges.

On the whole, security in grids is a particularly difficult problem. In global grid environments one must deal with diverse local mechanisms, support dynamic creation of services, and enable dynamic creation of trust domains (c. Def.2) [FOS98, FOS01]. Since global grid computing is

1 Some authors ad IT professionals prefer the term “Departmental Grid”, since “Cluster Grid” could be confused with “cluster computing”.

concerned with the sharing and coordinated use of resources and services scattered across multiple control domains, one must assume the absence of central location, central control, omniscience and existing trust relationships [GLO02]. These issues and constraints are of little if any concern in enterprise and cluster grids, but - in view of the considerations of the previous section - one here is faced with performance and interoperability issues.

2.1.Environmental requirements.

Common to all grid environments is the need to control resource sharing, by means of rules defining participants, resource providers, consumers, what is shared (also fraction of resources) and the conditions under which sharing occurs. By extending a definition introduced in [FOS98] for global grids, we refer to any set of individuals (figuring eventually as workgroups or institutions) and resources defined by the above rules as a *Virtual Organization* (VO)[2]. The sharing rules constitute the *VO policy* and the VO policy rules stating "who can do what and to whom" constitute the *VO security policy*. The entities involved in a VO security policy are distinguished in two classes, as *VO security subjects* and *objects*, respectively. A *VO security subject* is a grid entity that carried out an action under the control of the VO security policy. Besides of the specific grid considered, a security subject can be a user, a process acting on behalf of a user, a resource (such as a computer host, an application, a device, a file, etc.), or a process operating on behalf of a resource. A *VO security object* is instead a resource that is being protected by the VO security policy.

	Single user sign-on	*Coexistence with local security*	*Dynamic creation of services*	*Dynamic creation of trust domains*	*Support for transient groups of VO users*	*Support for secure group communications*
Cluster Grids	yes	yes, at the Operating System (OS) level	yes	yes, but not at the institutional level	yes	yes
Enterprise Grids	yes	yes, both at Cluster and OS levels	yes	yes, but not at the institutional level	yes	yes
Global Grids	yes	yes, both at Enterprise, Cluster and OS levels	yes	yes, coupling trust domains at lower levels	yes	yes

Since grids couple VO resources to enable advanced computing applications, their ultimate security objective is the execution of such applications according to the VO security policy. It turns out that grid security challenges are essentially related to *authentication* and *access control* in distributed systems and that a VO security policy must be a distributed *access control policy* designed on the basis of the peculiar environmental requirements for grids. These requirements somewhat vary depending of the tier of grid considered, as shown in table 1, and can be summarized as follows:

1. *single sign-on.* A user must be able to authenticate just once and then have access to all the grid resources on which he/she is authorized, even in case of using hundreds of distributed resources for a long period of time. Single sign-on increases easy of use, enabling to run complex applications without further user intervention. It relies on *delegation*, that is the facility to provide a process with the ability to run on a user's behalf, so that the process is able to access resources beyond those under the control of the (eventually distributed) operating system which hosts the user;
2. *Coexistence with local security.* Any grid environment is composed of multiple security domain, a security domain being a logical, administrative structure under the control of a security policy which is deployed and enforced by a single authority that the entities therein trust (c. Deff. 3-5). At the lowest level, a security domain is realized in a centralized way via an operating system, which implements all the communication channels and knows the user responsible for each

2 The meaning given here to the attribute " virtual" is slightly different than that in [FOS98], since in our case a virtual organization could in some case coincide with a concrete one.

process. An important requirement for grid architectures, which directly follows from its layered nature, is that grid security policies and mechanisms at one tier must only interact with those at the lower tier, without affecting them: as stated in the introduction grids are not constructed from scratch, and it will be impractical to modify both local resource access control policies and the mechanisms which implement them to accommodate inter-domain access; instead, one or more entities in a trust domain at tier *n* must act as agents of principals at tier *n+1* for local resources;

3. *Service creation.* Quite apart from the tier considered, a grid computation is composed of a time-varying set of processes running on different resources, possibly scattered among multiple security domains. A computation may acquire, starts processes on, and release resources dynamically during its execution;
4. *Domain creation.* According to the intentions of their designers [FOS98, FOS01, GLO02], global grids should be systems to be carried out on-the-fly, coupling previously existing trust domains into a wider one. Actually, because of reasons which we are going to investigate later, global grid security design suffers a number of scalability and overhead issues which hardly limit that requirement. In cluster and enterprise environments dynamic domain creation is a concern too, but only to accomplish *transient collaborations* (see below), since security policies are deployed and enforced on the basis of preexisting, real organizations;
5. *Transient collaborations.* As stated in [LOR04], many usage scenarios involving computational grids are actually based on small, ad-hoc working groups within a long-lived VO which need to establish transient collaborations with little or no intervention from the VO resource administrators. The two main kind of transient collaborations are as follows: (1) a group of users enrolled in the same VO which need to establish a collaboration in a peer-to-peer flavor, such that any member of the group can delegate some other group member to usage of resources (files, programs, disk space, CPU time, etc.) to which he/she has granted access by resource administrators, and (2) a user or a group of a VO which wish to delegate some other users or groups (not enrolled in the same VO, or not enrolled at all in any VO) to usage of the VO resources;
6. *Group communications.* The processes constituting a computation may communicate by using a variety of mechanisms, including multicast. A trivial solution for secure multicast is to set up a secure point-to-point connection between every two participants in the group communication session, but this solution is prohibitively inefficient in most cases and obviates the use of multicast routing [CAN99].

Grid environmental requirements address a number of security related issues that can be grouped into three main categories: *interoperability*, *expressiveness* and *scalability* issues. The first two kinds of issues are equally relevant for both global, enterprise and cluster grids; whereas scalability issues affect especially global grids.

Interoperability issues concern the interaction between, at one side, the security policies and mechanisms enforced at the operating system level and, at the other side, the VO security policy and the grid middleware used to enforce it. The major difficulty here is to cope with a multitude of operating systems, gaining a unique high level interface which is able to hide transparently their heterogeneity and the specifics of local implementations.

Expressiveness issues are related to the capability for VO users to have fine-grained access control to VO resources. Of course, that capability strictly depends upon legacy operating system commands and tools for accessing and managing system resources and for administering user privileges; what the above notion aims to convey in this context is that, with respect to the access and management of resources of any operating system S belonging to a given VO, VO users should not experience severe limitations if compared to standard users of S.

Security related s*calability* issues concern the overhead introduced to manage authentication, access control and user privileges in grid environments. Grids introduce the need to establish security relationships not simply in a client-server scenario, but in a more complex, hybrid programming environment which should encompass as much as possible peer-to-peer networks [FOX01,

LOR04]. In grid computations, potentially hundreds of processes distributed among hosts that collectively span many administrative domains must interact securely, and securely access to computing resources; furthermore, the dynamic nature of grids can make it impossible to establish trust relationships between sites prior to application execution.
Before investigating the specifics of these issues with respect to a grid security architecture, it is useful to report some concepts and terminology from the security literature. Moreover, it is clear that our discussion requires a formal definition for both cluster, enterprise and global grids. In turn, those definitions rely on a precise notion of *trust relationship* and related concepts, as *trust* and *security domain*. These topics will be covered in the next subsection and in §3, where we delineate a framework for grid security.

2.2.Security services for grids.
As previously stated, grid security is mainly targeted to the execution of applications according to a VO security policy. In view of the environmental requirements detailed in the previous section, that requires the control of resource sharing and usage in highly heterogeneous, dynamic, massive distributed systems. Moreover, grid applications require both correct addressing and integrity for application data and codes transferred over the grid to operate properly. As such, grid security concerns the design and implementation of:
a) communication protocols devoted to the *identification* of a VO security subject to a VO security object, and vice-versa;
b) policies and mechanisms for the deployment, administration and enforcement of privileges on the basis of the identification phase (a);
c) communication protocols which guarantee integrity and, eventually, encryption for data transfers among the VO entities which are granted to be involved in such transfers, on the basis of the privileges enforced by phase (b).

Items (a) and (c) are obviously related to entity authentication and message authentication. Item (c) is also related to confidentiality. Finally, item (b) consists precisely in the access control service.
The above services are not specific to the kind of grid considered; they are not specific to grids at all, since represent basic security services. With the only possible exception of the delegation facility (c. § 2, sign-on requirement), grids security indeed relays on standard services. What is specific to grids in general and to each one of the three tiers of grid, are the mechanisms to be used to realize such security services and the way those must be implemented. As we shall see, these issues strongly depends on the tier of grid considered.

2.2.1. Authentication
Entity authentication (or *identification*) is a service which assures one party (the *verifier*) of both the identity and aliveness of a second party involved (the *claimant*), and is at the basis of any access control mechanism. Entity authentication typically involves no meaningful message, but corroboration of a claimant's identity through actual communications with an associated verifier in real-time, that is while the verifying entity awaits. In grids, identification must be performed both in user-computer interactions than in computer-computer interactions. The latter typically occurs when a process running on behalf of a user on a given host needs a service or a resource from another host. In that case, authentication is required in both directions (*mutual authentication*), in order to guarantee both the host acting as server and the requesting process. Indeed, while users and processes are subjected to *impersonation* attacks, in a *spoofing* attack a system masquerades as another system, tricking a remote user or application into disclosing information.
Message authentication, where a message may represent both (a piece of) data and code, is instead a security service that provides to a party, which receive a message, assurance of the identity of the message's source (that is, the party which originated the message). This form of authentication typically provides no guarantee of timeliness, but involves meaningful messages and it is useful in circumstances where one of the parties is not active in the communication. It implicitly provides *message integrity*, since the receiver can verify if a message was modified during transmission, or substituted for a false one.

2.2.2.Confidentiality

Confidentiality or *secrecy* means to keep the content of information from all but those authorized to have it. This kind of service can be performed at various levels, and is specific to one of the two following security objectives: *channel protection* and *message protection*. In *channel protection* the goal is to avoid disclosure of both source, destination and distinctive traffic parameters to unauthorized parties. Instead, *message protection* aims to protect from disclosure the user data transmitted during a period of time. Message protection can be enforced at different degrees, depending on the percentage of protected messages and the protected fields for each message. Confidentiality should only be employed to transmit sensitive data, since the overhead it introduces could heavily influence the performance of the targeted application. For that reason, and since export control laws regarding encryption technologies are complex, dynamic and vary from country to country, confidentiality is only required as an optional service in grid environments.

2.2.2.Access control

In grids, *access control* is concerned with evaluating every request, submitted by a VO security subject, to access or use a VO security object to determine whether the request should be allowed or denied on the basis of the VO security policy. In this context the distinction between policies and mechanisms is important: *access control policies* define high-level guidelines establishing rules that regulate access to resources, whereas *access control mechanisms* are low-level software and/or hardware functions which act together to implement one or more access control policies. Typically, access control mechanisms are split in two functional moduli: *policy decision points* (*PDPs*) and *policy enforcement points* (*PEPs*), which may be co-located or separated. PDPs receive as inputs attributes and policies related to the subject who makes access request to a resource, along with resource specification, and output access control decisions; whereas PEPs enforce access decision made by PDPs. At the grid level, in view of the environmental requirement (2), access control mechanisms result in add-on software (and, eventually, in the hardware on which run such software) that acts as a "glue" with respect to the various access control mechanisms implemented at the operating system level. In grids, PEPs are realized at the operating systems level, reusing as much as possible legacy commands and tools for accessing and managing system resources and for administering user privileges. That is a natural consequence of the fact that resources are ultimately managed by operating systems. Instead, a typical approach followed in grid implementations is to realize PDPs in the middleware layer [LOR04]. Although that allows for more scalability in global grid environments, it could result in various security drawbacks; we are going to detail about this in a next section.

3.A framework for grid security.

As detailed in §2.2, entity and message authentication services are mandatory in grid architectures.
The most common technique used for both services is as follows: the verifier checks the correctness of a message (possibly in response to an earlier message), which demonstrates that the claimant (as entity or data-source, respectively) is in possession of a secret associated by design with the genuine party. Typically, the secret is a private or a symmetric key K and the proof of correctness consists in the verification that some data, encrypted with K, can be decrypted with a paired key K' for K, or vice versa[3]. In any case, practical implementations relies on *key management*, which establish how keying material is generated and how it is distributed, used and updated among authorized parties [MEN97] . Key management is an outstanding challenge in the context of grids, because such complex distributed environments require ad hoc communication models and procedures.
According to our definition of trust relationship (§3.1, def.1), one could equivalently say that key management concerns the way trust relationships are implemented and propagated between parties in a VO.

---

3 K' will be the paired *public* key in case of asymmetric cryptographic systems, and a key that can be easily deduced from K (possibly $K'=K$) in case of symmetric systems.

3.1.The notion of "trust", and related concepts.
In the sequel we give a formal definition for "trust" and for some related concepts. The concept of trust has a wide range of notions and aspects [SWA99]; the following definition was derived from a less formal one stated in [MEN97].

Def.1 (trust relationship) *Let* a,b *two entities. We say that* a *trusts* b *and use the notation* $a \vec{\tau} b$ *if one of the two following conditions occurs:*
i. a *and* b *share a key and/or a password* k *and* a *assumes that* b *shares* k *with* $c \in E$ *only if* a *shares* k *with* c*, too;*
ii. a *assumes to know an authentic and up-to-date public key of* b.

It easily follows that a trust relationship $\vec{\tau}$ satisfies the reflexive property. What about the symmetric and transitive ones ? We need to add some detail on "why" the involved parties are induced to do their assumptions, perhaps postulating something, so that one or both these two properties hold. For example, nothing can guarantee us a priori that, if (i) is satisfied, then b will be induced to do an assumption equivalent to that of a. Analogously, we cannot have evidence that if a assumes something on b and b assumes the same thing on c, then a will assume the same thing on c, too. This gap results because the previous definition does not say anything on how assumptions are built.
If both $a \vec{\tau} b$ and $b \vec{\tau} a$ hold, we say that a,b *mutually* trust each other and we write simply $a \tau b$.

Def.2 (trust domain) *Let* E *be a set of entities. Suppose that it exists a (mutual) trust relationship* τ *such that, for any* $e, e' \in E$*, it follows that* $e \tau e'$*. Then we say that* τ *is a trust relationship on the set* E*, and we call trust domain the set* E *considered with the above trust relationship* τ.

Def.3 (trusted third party) *Let* a *be an entity and* E *be a set of entities. Suppose that it exists a (mutual) trust relationship* τ *such that, for any* $e \in E$*, it follows that* $e \tau a$*. Then we say that* τ *is a trust relationship between* a *and the elements of* E*, or - alternatively - that* a *is a trusted third party (TTP) for the entities in* E*. We sometimes prefer the notation* τ_a *to* τ*, to stress that the trust relationship is enforced by* a.

Notice that a trust relationship τ between a and the elements of E is not in general a trust relationship on E, since $e \tau a$ for each $e \in E$ does not imply that $e \tau e'$ for any two elements $e,e' \in E$ such that $e,e' \neq a$; Conversely, it could happen that *any* such couple of elements e,e' is not in relation τ, because the transitive property does not hold for τ. In other words, a set E whose elements trust a given TTP a with respect to a trust relationship τ_a is not usually a trust domain under τ_a; we thus give the following:

Def.4 (trust domain generated by a TTP) *Let* a *be an entity and* E *be a set of entities such that* a *is TTP for* E*. We say that* E *is a trust domain generated by the TTP* a *if the set* E*, considered with the trust relationship* τ_a *among* a *and the elements of* E*, is a trust domain.*

Def.5 (security domain generated by a TTP) *Let* a *be an entity. Let* E *and* R *be a set of entities and a set of resources, respectively. A security domain under the control of* a *is a triple* (E,R,π)*, where* E *is the set of subjects and* R *is the set of objects of a security policy* π *which is deployed and enforced by* a*, that is a TTP for the entities in* E

Again, a security domain (E,R,π) does not results, generally speaking, in a trust domain for the set E of the objects of the security policy π, because π does not subtend a transitive trust relationship

between the TTP and E. In order to achieve such transitive property, we need to assume that a is not only a trusted third party, but it has the capability to give corroborate evidence of the trustworthy of a given entity to any other party: since two any elements in E trust a as the TTP of E, then they mutually trust each other. Besides of the reasons for that to occur, it is the fundamental, inherent characteristic of a *certification authority* (c. [MEN97]):

Def.6 (certification authority) *Let* a *be a TTP for a set* E *of entities. We say that* a *is a certification authority (CA) for* E *if, for any two* $e,e' \in E$ *such that* $e \tau a$ *and* $e' \tau a$, *then it follows that* $e \tau e'$.

3.2.Cluster, enterprise and global grids.

We said in §2 that grid security challenges are essentially related to authentication and access control in distributed systems, and that a VO security policy must be a distributed access control policy designed on the basis of the peculiar environmental requirements for grids. We said also that any grid environment is composed of multiple security domain, as per Def.5, and that, at the lowest level, a security domain is realized in a centralized way via an operating system, which implements all the communication channels and knows the user responsible for each process.

The main aim of this section is to investigate more deeply the authentication and access control problems in grids, detailing the above statement, and giving formal definitions for the three kind of grid environments which was informally introduced in §1.

At this point, for a better understanding of the sequel, it is useful to add some words on the way authentication and access control operates at the system level; that will lead us to remark the distinction between the notions of (registered) user, user name and subject (of a security policy).

In general-purpose operating systems (that is by far the usual case for grids), access control policies are usually *discretionary* [DOD85] in their nature, that is access to system resources is granted only on the basis of the *identity* of users and/or groups to which they belong[4]. Thus, user authentication is a prerequisite for access control and, if successful, it results in one or more user processes which inherit the user identity (UID) via suitable process identifiers (RUID, EUID, etc.). Modern operating systems implement various mechanisms to realize user authentication, but in any case those mechanisms are based upon some secret or unforgeable data which is shared among the operating system and the user[5].

Let s denote an operating system; then the access control implemented by it results in a security policy whose objects are the resources of s. Moreover, s is trusted, as per def. 1.i, both by the authority that specifies the policy and the users who are registered on s. The above trust relationship is mutual, since s assumes that a user shares its login and password with other users only if that information is related to a group account[6].

Thus s has a security domain (E,R,π) naturally associated to it (c. Def. 5), where π denotes the access control policy enforced by s, R is the set of resources belonging to s, and E is the set of entities which are subjected to that policy. E consists precisely in the time varying set of user's processes running on the system[7]. If U denotes the set of users registered on system s, and E is the set of subjects of the access control policy on s, then one of the tasks of the operating system, as a

4 Recent general purpose operating systems (e. g. Microsoft Windows NT and derivatives, Sun Solaris 10) provides Role-based access control (RBAC), in which authorizations to access system resources are assigned to roles, not to users anymore, whereas users are given authorizations to activate roles.

5 The shared information consists usually of the couple (login_name, password), where login_name is a public string uniquely associated by the system to any registered user, and password is a secret string owned by that user or, in more recent implementations, an hash of such string.

6 As a matter of fact, an operating system cannot detect if the password of a given user was compromised; we thus assume that passwords are good ones from the OS's perspective.

7 Although each access request to s is originated by users for the purpose of performing some action on the resources under the control of s, a user is a "passive" entity for whom authorizations are defined and who can only connect to the system. Thus, strictly speaking, users cannot be *subjects* of a security policy, since they do not carry out any action on a system (c. §2). Indeed, the operating system s does not interact directly with a user u to enforce the access control policy on s relative to u, but with processes running on s on behalf of u.

consequence of a successful authentication phase, is that of applying a mapping $\sigma: U \rightarrow E$, in order to properly associate user's identities to the activity on the system. Thus, it follows that $\sigma(U) \subseteq E$, not that $U \subseteq E$. However, in IT terminology the term *user* is often used in a broader sense than that conveyed by the expression *registered user*, indicating an accessing agent to a general object of a policy [PLF03]. In grid literature, for example, the term refers both to a person and an agent (that is, a process) on whose behalf the grid computation is being run [FOS98]. We adopt that terminology, denoting with U the set composed of both people which have an account on a given operating system s and agents acting on s on behalf of such people; thus, with a somewhat notational abuse, we speak in the following of "sets U of (registered) users such that $U \subseteq E$", where E denotes as usual a set of security policy subjects.

A one more distinction takes relevance in our analysis of grid security, that concerning *registered users* and *user names* (or, equivalently, *login names*). It could appear somewhat obvious and useless to remark the difference between these two notions; indeed, when it is clear from the context, we sometimes use the term *user* purporting instead as *user name*, for conciseness reasons. What we wish to stress here is that our model has to emphasize in some circumstances that difference. First of all, differently than in the single system case, where usually one has a one-to-one mapping from the system's users to the user's identifiers for that system, in the context of grids the usual case is that to the same grid user correspond many user names, each name being the identifier of that user for a particular system. Thus, our model needs to sharply differentiate between individuals (or groups) which figure in a VO and the way such individuals are registered on the various operating systems that belong to the pool of resources that such VO has access to. Moreover, the same individual can figure on a system s both as a user of s and a user of a grid which includes s among its resources. Eventually, both the previous circumstances could be simultaneously true. Because of the environmental requirements (1) and (2), that is *single sign-on* and *coexistence with local security* (c. §2.1); then it follows that to avoid conflicts one has to distinguish between identification at the operating system level and identification at the various grid levels (that is: cluster, enterprise and global). For example, suppose that there are two users u_1, u_2, which are registered with the same user name u' on systems s_1 and s_2, respectively. Suppose now that user u_1 gets enrolled in a grid which includes s_1 and s_2 among its resources. If u_1 had the same identifier u' both at the operating system and grid levels, it would be impossible to distinguish u_1 from u_2 on system s_2. The need for each VO user to have two kinds of user name, a VO user name and a potentially different local name on each VO resource was clearly expressed in [FOS98]. In our tiered approach to grid security, the general case results in a user u which has a single global user name to access as user of a global grid, plus as many user names as many enterprise grids, cluster grids and operating systems compose the global grid[8]. Each level requires its separate name space, and an user access at one level requires as many name translation as many tiers the user must traverse to reach that level. For example, suppose that a user u, enrolled in an enterprise grid Eg, wishes to read a file under the control of an operating system s, which belongs to a cluster environment Cg embedded in Eg. That access requires a total of two name translations: a first one from the Eg to the Cg name space, and a second one from the Cg to the s name space. Of course, if s was included in Eg directly as a system and not as a Cg component, it would suffice only one translation from the Eg to the s name space. The mapping of a name space to another one has to be defined at the lower level, and is site-specific. For example, the Cg interface at Eg could map Eg names to a predefined Cg name, a dynamically allocated Cg name, or a single "group" name. That is again a consequence of requirement (2), which poses also a somewhat different restriction: if a user u has an account u' on a system s, and then he/she gets enrolled in a grid which includes s among its resources; then he/she should be logged on s as u', instead of being logged with the user name(s) that the grid interface on s uses to login grid users that do not have a specific local account on s.

All that premised, we assume that a grid is an operating environment (U,S) composed of a finite set

8 It should be clear that, because of requirement (2), a naming convention such that the one used in the *Domain Name Service* cannot be successfully employed here.

R of resources distributed among a finite set S of (possibly distributed) operating systems, and that the resources in R must be shared among a finite set U of users in such a way that:

1. Any resource $r \in R$ is under the control of just one $s \in S$; i. e. $R = \cup_{s \in S} s$, where $s \cap s' = \emptyset$ if $s \neq s'$. For commodity purposes, we enumerate the elements in S as $s_1, s_2, ..., s_m$, so that $s_i = \{r_{i1}, r_{i2}, ..., r_{ij(i)}\}$ denotes the operating system in S to which pertains the grid resources shown in brackets;
2. The system $s_i = \{r_{i1}, r_{i2}, ..., r_{ij(i)}\}$ enforces a certain access control policy π_i on a set U_i of users, which are enrolled in the grid environment and are registered on system s_i. That, as previously stated, results into a security domain (E_i, s_i, π_i), where $U_i \subseteq E_i$ and E_i consists precisely in the time varying set of user's processes running on system s_i;
3. A grid user $u? U$ is subject to access control policies and mechanisms which inter-operate with the ones enforced at the system level, in order to assure that grid resources are used properly;
4. Systems and users communicate along *unsecured channels*, from which parties other than those for which the information is intended can reorder, delete, insert, or read;

It follows that (U,S) results in m security domains (E_i, s_i, π_i), which control accesses by (possible) overlapping sets of users to disjoint sets of resources, on the basis of security policies that are specific of the system to which the requested resource belongs.
Requirement (3) above corresponds to the *coexistence with local security* requirement and, together with the *single sign-on facility*, it results in the following two coexistence statements:

Statement.1 (Coexistence with previous enrollments) *Let* (U,S) *be an operating environment in which a set of users* U *is entailed to use a set of resources* S. *Suppose that* $S = \cup S_i$, *where* S_i *denotes the set of resources related to the operating environment* (U_i, S_i), $(i = 1, ..., n)$.
A global naming scheme for (U,S) *is a 1-to-n mapping* $\gamma: U \rightarrow \cup U_i$ *such that:*

- $\gamma = (\gamma_1, .., \gamma_n)$, *where* $\gamma_i \in U_i$ *for each* i *;*
- *if* $u \in U$ *is a user name related to a user which was previously registered as* u_i *in* (U_i, S_i), *then* $\gamma(u) \cap U_i = u_i$.

Statement.2 (Coexistence with previous domains). *Let* (U,S) *be an operating environment as in statement 1, and let* $\gamma: U \rightarrow \cup U_i$ *be a global naming scheme for* (U,S). *Let* (E_i, S_i, π_i) *be the security domain associated to the environment* (U_i, S_i), *where* π_i *denotes the access control policy local to* S_i, E_i $(E_i \supset U_i)$ *is the set of entities which are the subjects of that policy, and* U_i $(E_i \supset U_i)$ *is the set of users entailed to use the resources of* S_i.
If an user u *requests access to a resource* r, *an (overall) access control policy* π *for* (U,S) *acts as follows:*

- *if* $u \notin \cup U_i$ *or* $r \notin \cup S_i$, *then* π *rejects the request;*
- *if* $u \in U_i$ *and* $r \in S_i$, *then* π *acts as* π_i *;*
- *if* $u \in U_i$ *and* $r \in S_j$, *with* $i \neq j$, *then* π *uses* γ *to map* u *to* $u' \in U_j$, *eventually imposing some access restrictions on* u' *as acting on behalf of* u, *and finally acts as* π_i.

It is important to remark, at this point, two basic facts about the grid access control policy π introduced via the previous statements.
First, analogously that access control policies at the operating system level, π has been conceived as a discretionary access control policy; this is probably the simplest way to satisfy the *coexistence with local security* requirement and was a main design choice for the Globus Toolkit [FOS98]. However that choice results in serious drawbacks in the implementation of global grids, since it imposes a trade-off between *expressiveness* and *scalability* issues. Indeed, if $u \in U_i$ and $r \in S_j$ with $i \neq j$, then π relays on *impersonation*: the user u is mapped to another user u', which is a subject for the access control policy local to S_j. Thus, the finest-grain access control in (U,S) is achieved when the global naming scheme γ is partially injective on any U_i, that is if from $u, u' \in U$ and $u \neq u'$ it then follows that $\gamma(u) \cap U_i \neq \gamma(u') \cap U_i$. But this case results, for any operating system belonging to a

given VO, in a number of local accounts which is at least equal to the total number of the VO security subjects. Since any new account requires (manual) intervention from system administrators, the above scenario has the drawback of daunting management and it is not scalable at all, so it can be employed only for grids of small dimensions (that is, with a small set of users and/or resources).
The second thing we wish here to emphasize about the grid access control policy π introduced previously, is that its effective enforcement requires that entities and systems belonging to different security domains are mutually authenticated. Indeed, if an entity $e \in E_i$ requests access for a resource $r \in S_j$ with $i \neq j$, then:

- avoiding *impersonation attacks* [PFL03] requires that S_j has corroborated evidence that the system from which the request was issued is actually S_i, and that e is not impersonated by another entity on S_i (in general, granting access to a resource depends both on the remote system from which the resource usage request was issued and the particular user which made the request);
- avoiding *spoofing attacks* [PFL03] requires that e has assurance of the identity of S_j.

Thus it follows that both users and systems related to a grid environment must all belong to a trust domain E, as per Def.2. Eventually, E could be a trust domain resulting from the aggregation of multiple trust domains, and in such case we shall refer to E as a *composite* trust domain. In both cases (simple or composite), it must exist a set of communication methods and procedures such that any two entities in E can mutually trust each other; these communication methods and procedures can be organized in a variety of ways, which are called *trust models*.
As we are going to show, one main difference between cluster, enterprise and global environments consists actually in the nature of E as a trust domain.
Another major difference, as stated in the introduction, concerns Policy Authorities: in cluster and enterprise grids system-wide access control policies are defined, implemented and managed with respect to a *root* Policy Authority, whereas that is not the case for global grids.

Def.6 (Cluster grid) *A cluster grid is a grid environment* (U,S) *with the following additional properties:*

5. *Any* $s_i \in S$ *represents an operating system with an associated security domain* (E_i,R_i,π_i)*, where* $U=\cup U_i$*,* $U_i \subseteq E_i$ *and* E_i *consists precisely in the time varying set of user's processes running on system* s_i*;*
6. *There is an overall access control policy* π*, satisfying the coexistence statements with respect to the security domains* (E_i,s_i,π_i) *associated to the operating systems* $s_i \in S$*;*
7. π *is defined, implemented and managed by a single Policy Authority;*
8. *Users and systems are interconnected via a local area network, such that the overhead introduced by any cryptosystem is not negligible with respect to the network latency.*

As directly follows from requirement (5), any cluster grid environment (U,S) involves a trust domain (E,τ), where $E=U\cup S$ and τ is a trust relationship such that $p\ \tau\ q$ for any p and q, where p, q may denote both users involved into the grid, processes running on behalf of such users and operating systems $s_i \in S$. Since, typically, an operating system is not a certification authority for the users processes running on the system, the security domain (E_i,s_i,π_i) is not a trust domain, so that E results in a simple trust domain.

Def.7 (Enterprise grid) *An enterprise grid is a grid environment* (U,S) *such that* $U=\cup U_i$, $S=\cup S_i$ *and with the following additional properties:*

5'. *Any* (U_i,S_i) *represents a cluster grid as per def. 6, with an associated security domain* (E_i,R_i,π_i)*, where* E_i *results in a trust domain with respect to the trust relationship induced by the access control policy* π_i*;*
6'. *There is an overall security policy* π *satisfying statements 1 and 2 with respect the security domains in (5');*

7'. π *is defined, implemented and managed by a single Policy Authority;*
8'. *Users and systems from different clusters are interconnected each other not necessarily via a local area network, and the overhead introduced by suitable cryptosystems is negligible with respect to the network latency;*

Def.8 (Global grid) *A global grid is a grid environment* (U,S) *such that* $U=\cup U_i$, $S=\cup S_i$ *and with the following additional properties:*
5''.*Any* (U_i,S_i) *represents an enterprise or a cluster grid as per deff. 6 and 7, with an associated security domain* (E_i,R_i,π_i), *where* E_i *results in a trust domain with respect to the trust relationship induced by the access control policy* π_i*;*
6''.*There is an overall security policy* π *which satisfies the coexistence statements with respect the security domains in (5'');*
7''.π *cannot be defined, implemented and managed by a single Policy Authority; instead, that requires the mutual interaction of as many Policy Authorities as the enterprise and cluster grids which compose the global one;*
8''.*Users and systems from different* (U_i,S_i) *are interconnected each other via the Internet, and the overhead introduced by any cryptosystem is negligible with respect to the network latency;*

In enterprise and global grids the related trust domains are composite; they, indeed, are obtained in a recursive manner, by imposing suitable trust relationships among lower level trust domains in such a way that both coexistence statements 1 and 2 are satisfied.
It is useful to remark here that the global access control policy for any grid environment of tier *n* (*n =1,2,3*) results in the access control policies enforced at the operating system level, plus *n* authentication schemes whose function is solely to guarantee the proper application of such local policies in the VO's context. Moreover, as implicitly stated Statement 2, some access control could be performed at some grid level, and eventually for all tiers of a given grid.

3.3.The reference model.

It should be clear from what stated in the previous section, that there are three distinctive and strictly interlaced characteristics in a grid security architecture:
- the way authentication and authorization is realized at the operating system level;
- how (if any) authorization operates at the grid level;
- how authentication schemes are implemented in the different grid tiers;

Let us consider a grid environment of tier *n* (*n =1,2,3*); for the description of its security architecture, it is useful to distinguish three different logical subsystems: the *Virtual Organization (VO) space*, the *Trust (T) space* and the *Process (P) space*, as shown in Fig.1.
The VO space is the framework in which VOs are deployed and organized; it consists in the set of entities which are the subjects or the objects of the grid access control policy π at tier *n*, and in π itself, which ultimate goal is to control accesses of grid processes originated from VO's security subjects to the VO's security objects (c. §3.2).
The VO is composed of multiple security domains, each security domain being a set of users and resources which are under the control of the same operating system. Such security domains have their own local access control policies π_i, and they are glued together by means of the T space into the wider trust domain which is at the basis of the overall access control policy π. The T space consists, in turn, of the set of communication methods and procedures such that any two entities in the VO can mutually trust each other; and the way those methods and procedures are organized constitutes the trust model adopted in the grid. Because of the tiered nature of grids, the T space actually consists of different trust domains; in such a way that at the highest tier *n* corresponds only one trust domain and that if a domain related to tier *p* is embedded into a domain of tier *q* then $p<q<=n$.
With respect to the above trust models, it first comes out that - as for any other complex distributed environment - public-key techniques offer relevant advantages versus symmetric-key ones,

allowing simplified management, better scalability and enhanced functionality. Indeed, asymmetric techniques offer "built in" features such as non repudiation and single-source data origin authentication; moreover, as opposed to secret keys, only authenticity of public key is required, which can be achieved by means of a trusted off-line service performing key authentication plus an untrusted on-line server for delivering such authenticated keys to users [MEN97].

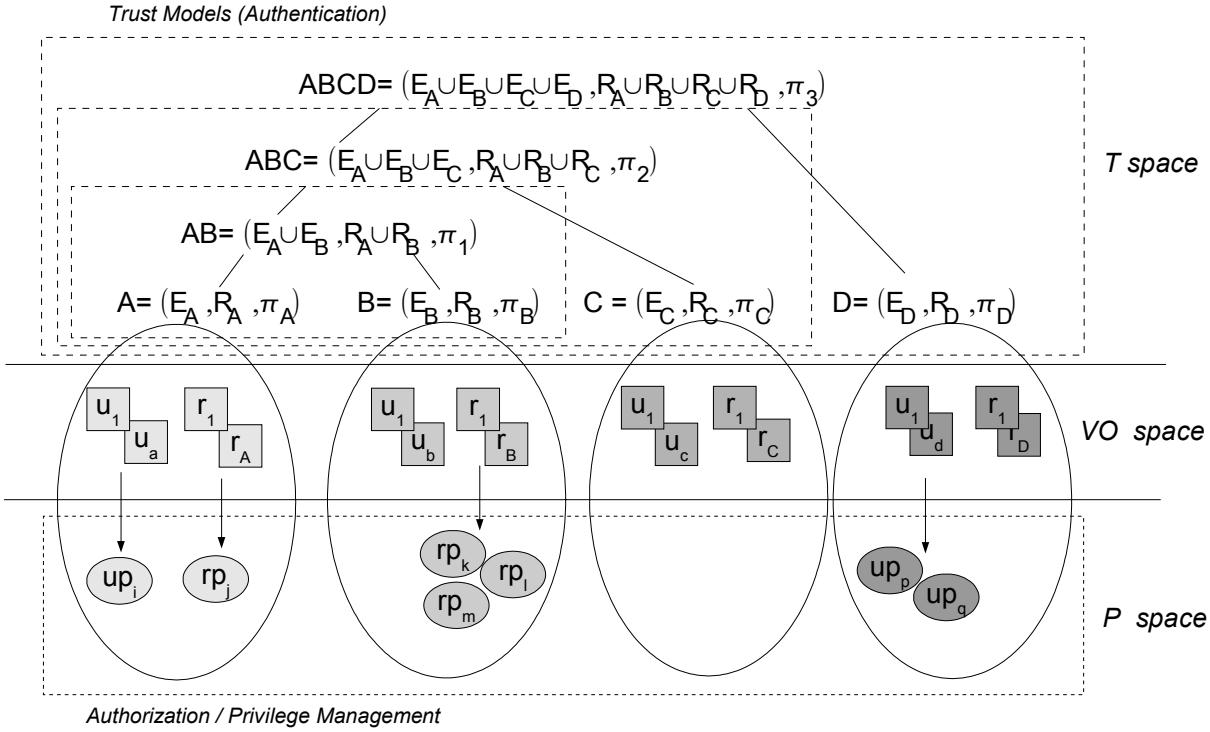

Fig. 1 – *Two logical subsystem, the T (Trust) space and the P (Process) space surround the VO space and are related to grid authentication and access control, respectively. The figure illustrates a special case in which four operating system security domains A,B,C and D are glued together in the T space in such a way that A and B give rise to a cluster grid AB, AB and C result in an enterprise grid ABC and, finally, ABC and D realize a global grid.*

Thus, a public key infrastructure (PKI) is mandatory in the T space, and the trust models above result in some of the existing alternatives to provide a "chain of trust" from a public key that is known to be authentic through to a specific user's or resource's public key. PKIs are usually deployed and implemented with respect to the X.509 Internet standard; this is the rule of thumb for grids too [FOS99, GRI99], so in the next subsection we will focus on analyzing X.509-based public-key infrastructures (PKIX).

The last subsystem is the P space, and relates to process management (e.g. : creation, process-to process interaction, access to resources) in the context of a grid computation. From a security viewpoint, the P space is concerned with authorization and management of user and process privileges. This subsystem assures the proper execution of grid aware applications with respect to the parties and the resources involved, whose identity is corroborated on the basis of the authentication service implemented in the T space. In this context too, because of the nature of grid computations, grids demands for tailored security protocols and mechanisms; but, conversely than in the T space case, now there are not reference standards to adopt.

3.4.The X.509 Internet standard.

The X.509 Internet standard is a subset of the ITU-T X.500 specifications for a *directory service,* that is a (possibly distributed) public available database about a set of users and some related information (such as their network addresses, public-keys, and so on). X.509 concerns the definition of authentication procedures which are based upon the notion of *public key digital*

certificate (or *digital signature*). The use of public key certificates (PKCs), as proposed in [KOH78], was to securely bind a user's name to his public key, thus providing a means of entity authentication. This use was subsequently adopted by X.509 v.1 certificates [X509v1] and Pretty Good Privacy (PGP) certificates [ZIM95]. The application receiving such a certificate could answer the question "who is wanting to access the service", concerning authentication, but could not answer the supplementary question related to authorization or access control, that is "which (level of) service is this user allowed to access", without recourse to additional, external access control information. The nowadays up-to-date version 3 of X.509, issued in 1995, provides some optional extension fields for PKCs, allowing information related to both authentication and authorization to be contained in certificates [X509v3].
An X.509 PKC [X509v3] is issued by an entity a which guarantees about the correct binding of a given user (the subscriber) to his public key. It is basically an electronic document consisting of a data part and a signature part. The data part contains clear-text information about the identity of the subscriber (a multi-component, alphanumeric string called *Distinguished Name* or briefly *DN*) and its public key[9], plus other (sometimes optional) miscellaneous information such as additional knowledge (e.g., street or network address) about a, the lifetime of the certificate, its serial number, additional information about the key (e.g., algorithm and intended use), etc. The version 3 certificate contains some more extension fields which enable to bind public keys to attributes of an entity other than the entity's identity, such as a role, a title, a creditworthiness information or – as stated before – an access control information with respect to one or more resources. The signature part consists of the hash of the data part encrypted with the a's private key. If e is a party and a issues a public key certificate for it, then X.509 uses the notation $a\langle\langle e \rangle\rangle$ to indicate such certificate. The issuer a is supposed to have, with some minor variations that does not influence our reasoning, the following behavior:

C1. manages its private key with integrity and confidentiality protection, assuring that no other entity can sign certificates with that key;
C2. implements a secure, peer-to-peer channel with each principal in a set E, which represents the set of users that are authorized - as subscribers[10] - to make use of the service offered by a;
C3. delivers its public-key(s) to each $e \in E$ and receives e's public-key(s) via the secure channel implemented in the previous step;
C4. issues public-key certificates for its subscribers;
C5. revokes a user's certificate, and issues a new one containing the new replacement public key, in case that a user's public key becomes comprised, or is lost or stolen;
C6. manages the list of revoked certificates (Certificate Revocation Lists, CRL), updating and signing it regularly to guarantee a valid and up-to-date information;

Because of C1-C3, it follows that (E,τ_a) is a trust domain generated by the TTP a (c. Def.4), where τ_a is the relationship defined in the set E, as follows:

$$\tau_a\colon\ e\ \tau_a\ e' \quad \textit{iff} \quad a\langle\langle e' \rangle\rangle$$

Indeed, τ_a represents a trust relationship in the sense of Def.1.ii., and precisely a mutual trust

9 There are, however, some exceptions to the above binding among a subscriber user-name and a public-key. Two of them include devices certificates, in which the subscriber is usually the individual or organization controlling the device, and anonymous certificates, in which the identity of the individual or organization is not available from the certificate itself.

10 The above secure channel is provided to a user as a consequence of a *registration phase*, which requires itself verification of the identity of the user and its enrollment as subscriber of the service with a DN within a name-space for the subscribers of the a's service. Those previous two functions may be delegated to entities different than a, in which case we will have a separate registration authority and name server; conversely, a may carry out both the function of a registration authority and a name server. In any case, the registration and naming functions have to be performed in such a way that the DN and the public-key are permanently assigned to only one entity: even after revocation or expiration, they cannot be reused for another entity.

relationship on E, with a as a TTP for E. Moreover, since a public-key certificate is a way to securely bind a public-key to the identity of a subscriber, and because of C4-C6, it follows that a constitutes a certification authority for E, in the sense of Def.6. On the basis of the previous result, one can implement PKIX-based mutual authentication among principals in E, as explained in the following. Before describing the authentication protocol itself, we need some detail on how public-private key pairs are managed by principals. Key management for any given principal $e \in E$ proceeds in the following two steps:

K1. The principal e generates a public-private key pair and stores the private key in local storage with integrity and confidentiality protection. It sends its public key to the certification authority a, using the secure channel implemented by a in step C2;
K2. The public-key of e and its identifying information (e's distinguished user name) are signed by a, resulting in the so called e's public-key certificate $a\langle\langle e \rangle\rangle$ issued by a, which is commonly returned to e.

Finally, authentication of a principal $e \in E$ (called in such respect the claimant) to another principal $e' \in E$ (the verifier), proceeds as follows:

A1. the verifier e' obtains the public key of the certification authority a, using the secure peer-to-peer channel implemented for it by a in step C2, and stores that key with integrity protection. Then e' sends to the claimant some (random) data D, as a challenge of identity and aliveness (c. §2.2), storing it locally with integrity protection, too;
A2. the claimant e signs the data D with its private-key; then sends the resulting signature on D and its public-key certificate $a\langle\langle e \rangle\rangle$ to the verifier e';
A3. the verifier uses a's public-key and $a\langle\langle e \rangle\rangle$ to obtain evidence of the authenticity of e's public-key, then decrypts by means of that key the received encrypted D from e. If the output of the above procedure is D, then e' assumes to be in communication with e, otherwise not.

In order for a certification authority to communicate with its subscribers in a cost-effective and reliable way, because of (C2), there must be an existing close relationship between them. Generally, authorities only have a suitable relationship with a limited community; on the other hand, like in global grids, PKIXs are often required to scale beyond such limits. Therefore, PKIXs with multiple CAs are required, and the authorities' keys must be shared amongst the participants in a secure manner: the way of that secure sharing determines the adopted trust model in the PKIX.
The X.509 standard does not insist on any particular trust model [X509], since as an international standard it should be able to cater for all requirements; it only establishes the following one more capability for an issuer:

C7. a CA can eventually manage trust relationships with other CAs;

and introduces the notion of *certificate chain* as the basis to propagate trust from a public key that is known to be authentic through to a specific user's or resource's public key.
Let e, e' be subscribers for authorities a and a', respectively; if there exist a sequence $a=a_1, a_2, .., a_n=a'$ of CAs by means of those it is possible to establish a chain of X.509 certificates, then e trusts e':

$$e \ \tau \ e' \quad \text{if: } a_1\langle\langle a_2 \rangle\rangle a_2\langle\langle a_3\rangle\rangle \ldots . a_{n-1}\langle\langle a_n \rangle\rangle$$

where τ is again a trust relationship in the sense of Def.1.ii, which has to be considered as an extension of τ_a beyond the set E. The above result is a straightforward consequence of the fact that $a_1, a_2, .., a_n$ are all certification authorities in the sense of Def.6.
Because of Internet's massive growth and recent developments in theory and practice of complex

distributed systems (such as computational grids), the X.509 standard is actually quickly evolving, especially with respect to certificate formats and types. It is possible to recognize two main trends in this evolution.
The first one can be traced back to some preliminary work developed at the IETF SPKI working group, which addressed the issue of name-key bindings in PKCs and realized that such certificates were of extremely limited use for the emerging needs of the Internet community as respect to trust management [RFC2692, RFC2693]. Versions 1 and 2 of the X.509 standard, issued in 1988 and 1993 respectively, provide only for PKCs, whereas version 3 introduces *attribute certificates* (ACs). ACs have a structure similar to PKCs, but are not intended to provide authentication, and for such reason contain no user's public keys. The main motivation for the introduction of X.509 ACs is that the placement of authorization information in PKCs is usually undesirable because of the two following reasons:

- authorization information usually have a shorten lifetime of authentication information;
- the PKC issuer is not usually authoritative for the authorization information.

ACs may contain attributes that specify group membership, role, security clearance, or other authorization information associated with the AC holder; they are intended to represent a suitable token for role-based access control decision functions. Yet, when making an access control decision based on an AC, it may need to ensure that the AC holder is the entity that has requested access; thus ACs incorporate a reference to a PKC for the AC holder. ACs are currently used in GT4 implementations.
The second trend in the evolution of the X.509 standard refers to the introduction of a further new type of certificates, called *proxy certificates*, to realize identity delegation (impersonation) in the Globus environment (c. §3.2); since proxy-certificate profile is nowadays only a proposed Internet standard [RFC3820], we will discuss about it in the next section, devoted to the security architecture of Globus.

4.The Grid Security Infrastructure.
The main aim of this paper is to establish where and how current grid security technologies fail to meet the environmental requirements illustrated in §2.1, indicating possible alternatives and areas of future research. In deploying our analysis, we choose the Globus Toolkit (GT) [GLOB] as a concrete grid implementation of reference, since it represents the de facto standard for global grids. Thus, it takes relevance to state beforehand a brief description of its security architecture, the *Grid Security Infrastructure*.
The *Grid Security Infrastructure* (GSI) implements all the security services discussed in §2.2, enabling cross-domain mutual authentication, message integrity protection, (optional) message content protection and access control to VO resources from VO users. Actually, there are different versions of the Globus Toolkit (from the oldest GT 1, introduced in 1998 [FOS98, GLOB], to the current GT 4 version [,GLOB]) and, with the only exception from GT 1 to GT 2, GSI has evolved at every GT upgrade. These three GSI versions differs both in the provided services and in their implementation, and in the following we refer to them as GSI 1-2, GSI 3 and GSI 4, respectively. Despite of these differences, however, all GSIs are built upon the *Transport Layer Security* (TLS) protocol, an IETF proposed standard for end-to-end security over TCP/IP derived from SSL v.3 [RFC2246]. SSL (TLS) authentication services makes use of a PKIX, and all GSIs adopt a proprietary mechanism (which can slightly vary from one version to another) to perform *identity delegation*. GSI 3 is based on and largely compatible with GSI 1-2, but has been improved (especially in the delegation mechanism; c. §4.1.1) and augmented to be compliant with the Web Services (WS) approach [W3CWS]; indeed, it supports the *Simple Object Access Protocol* (SOAP) [BOX00], providing context establishment, message authentication and message confidentiality.
Like GSI 3, GSI 4 provides distinct WS and pre-WS authentication and authorization functionalities for backward compatibility with GSI 1-2. Again, GSI 4 has been updated and augmented with functions aimed at securing a services based architecture: its main new features with respect to GSI

3 are the compliance with WS emerging standards, and the introduction of the *Delegation Service* (see below).

4.1.Authentication

As previously stated in §3.2, *identity delegation* (or *impersonation*) is the way adopted in Globus to enable computations to be spawned among multiple trust or security domains. Processes that constitute a Globus computation are fundamentally of two kinds, and are respectively designed as *user proxy* and *resource proxy*. A user proxy is a session manager process given permission to act on behalf of a user for a limited period of time; analogously, a resource proxy is a process that has permission to act temporarily on behalf of a resource [FOS98]. It could be useful to remark here that the notion of user proxy in the context of a grid application is somewhat the counterpart to that of user process in the case of standalone operating systems. User and resource proxies were introduced as suitable interfaces among VO' s components, which make possible to carry out their mutual interactions during the execution of grid applications, according to the environmental requirements depicted in §2.1, first of all the *single sign-on* facility and the *interoperability with local computing environments*.

Delegation is realized via the *Globus delegation protoco*l and the *Globus proxy certificates*. The Globus delegation protocol provides a process (called *acceptor*) the ability to run on behalf of another entity (called *initiator*), so that the acceptor can make requests to a third party as for the initiator. It is intended to be an add-on to the set of TLS protocols which operate upon the Record Layer [RFC2246]. The protocol can be started by both the initiator and the acceptor and, in case the acceptor agrees to satisfy the initiator request, it ends up with a delegating identity token transmitted from the initiator to the acceptor. The token can be based on both Globus proxy-certificates and Kerberos forwardable tickets [KOHL94]. A proxy-certificate (PC) is an X.509 v3 PKI certificate with the following properties [RFC3820][11] :

- its issuer is another PC or a X.509 *End Entity Certificate* (EEC), that is an X.509 PKC issued for an end entity, such as a user or a service, by a CA ;
- it can sign just another PC and cannot sign an EEC;
- it possesses its own public-private key pair;
- its DN is obtained by appending a Single Common Name component to the DN of the initiator, in such a way that it has a unique identity derived from the identity of the EEC;
- it has one more extension field that indicates that a certificate is a proxy certificate and whether or not the issuer has placed any restrictions on its use.

To create a proxy-certificate, If the PC request is valid, then a PC signed by the private key of the EEC or by another PC is created. When a PC is created as part of a delegation from an initiator A to an acceptor B, then the following actions are performed as part of the delegation protocol:

- B generates a new couple of public-private keys and uses such keys to create a request for a PC that conforms to the certificate profile sketched above;
- B passes the PC request to A over an authenticated, integrity checked channel;
- A verifies that the requested PC is valid and, if that is the case, the PC signed with A's private key is returned back to B. To be valid, a PC has not to be an ECC and its fields must be appropriately set (in particular, the binding between the PC public key and its DN has to be correct).

It is useful to remark here that a proxy certificate is generally less secure than the ECC that issued it; indeed, for single sign-on purposes the private key of a PC has to be stored unencrypted, so that it is protected only by file-system security mechanisms. Anyway, because of the delegation

11 Actually, GSI 3 and 4 support two kinds of PC: the old Globus proxy-certificate (introduced with GT 1) and the RFC3820 compliant proxy-certificate (introduced with GT 3), which is a revised version of the first one and, as previously stated, is nowadays a proposed Internet standard, which is intended to be an extension to X.509 [RFC3820]. The main difference between these two PC formats is that the new one supports the use of policy statements to restrict the set of delegated privileges. Other than enabling a more selective use of privileges, that improves security in case of a stolen proxy certificate.

protocol design, neither A gets knowledge about B's private key nor B about that of A. Moreover, because of the PC validity checks performed by the initiator A in step 3 above, a PC cannot be used to sign an ECC or PC for a user different from A. Further, a compromised PC can only be misused for a single user's privileges, and within the bounds of the lifetime and the restriction policy which are set up for that PC.
Pre-WS (that is, GSI 1-2) authentication services are implemented via the GSS-API [RFC2743, RFC2744], extended with the functions described in the GSS-API Extensions document [RFC3820]. The GSS-API is a generic API for doing client-server authentication, based upon SSL/TLS protocol, and [RFC3820] extend the standard path validation mechanism to handle proxy-certificates and X.509 extensions (c. §3.5). Below the GSS-API layer there are multiple APIs which perform various operations upon which the GSS layer relays. The general design principle guiding these APIs is data encapsulation: the state of the system is captured via suitable data structures (handles and attributes), which are then acted upon by various setters and getters, as well as other functions.
GSI 3 implements SOAP with XML-Signature [BAR02] and XML-Encryption [EAS02] for authentication and message protection, and *WS-SecureConversation* protocol [DEL02] for context establishment.
In GSI 4, authentication and authorization message-level security was updated to reflect both *OASIS standard* 1.0 [OASIS] for WS Security and published IBM/Microsoft specification for *WS-SecureConversation* [WSSC1.1] (WS-SecureConversation implementation released with GSI 3 is now deprecated). Moreover, GT 4 provides the *Delegation Service*, a new component designed to act as an interface for delegation of credentials to an hosting environment and the management of such credentials. This service enables a single delegated credential to be shared across multiple invocations of services on the hosting environment, and provides a means for credential renewal. The Delegation Service is built upon the TLS protocol, the Globus delegation protocol and the proxy-certificate profile [GT4SD].

4.2.Authorization.
GSI adopts different mechanisms to accomplish resource authorization, including a custom authorization handler, a *grid-mapfile* access control list, an access control list defined by a service and access to an authorization service via the SAML protocol.
The pre-WS security framework provides two authorization APIs: the *generic authorization* API and the *gridmap* API. The generic authorization API allow developers to implement custom authorization modules. The gridmap API provides a default authorization and mapping mechanism based on a local file, called the *gridmap file*, but allows also for custom callouts to be plugged in and override the default behavior. The gridmap file maps Globus identities to local user identities: it is a simple ASCII file in which each row contains the distinguished name of a Globus user (as reported in the DN field of the X.509 PKC used to authenticate that user; c. §3.3), and a valid user-name for the local system, separated by one or more blank spaces.
WS-style authorization framework include an access control list defined by a service and an access to an authorization service via the SAML protocol.
All the above mechanisms require that a Globus user is registered as a standard local user on an operating system `s` to access a resource which results under the control of `s`. As we detailed in §3, that is not a limitation of the Globus approach, but an unavoidable consequence of grid's environmental requirement of coexistence with local security. The advantage that GSI can be layered on top of existing operating systems without undermining their local security mechanisms comes at high price: that of constraining access control *expressiveness* and *scalability* on the size of the VO to be mutually exclusive alternatives. Various mechanisms have been proposed to mitigate that drawback, but the only one included in the GT distributions (starting from version 2) is the *Community Authorization Service*.
The Community Authorization Service (CAS) is intended to reduce administrative overhead by separating VO's administration from specific resources' administration. With CAS, resource's administrators can grant bulk rights to communities (which can coincide with an entire VO or be

smaller groups inside a given VO), and CAS administrators then decide what subset of a community rights an individual member will have. Group members authenticate to grid resources with a group credential which has restrictions applied to it narrowing the individual's rights to a subset of the rights the community has at the resource. The limitations imposed on the rights the group account has at the resources are enforced at the grid level by the Globus application. In CAS, the requirements for existing local user accounts is eased through shared group accounts and the community is given the flexibility to manage membership and member's privileges without resources administrators. However, this approach has three main drawbacks. The first is that it violates the least-privilege principle [SAL74], since CAS enabled services execute with wide access rights and need to self-restrict their use of the underlying resources. Second, it requires enforcement of authorization and access control within the grid middleware, thus creating a need for trusted application code. Third, it is based on a preexistent group owned infrastructure component (the CAS server) and a community administrator.

5.A tiered approach to grid security.

If, on the basis of what detailed in the introduction, one agrees in coming to the conclusion that it is useful to design and implement grids as tiered objects, then the same should be applied to their security architecture; that is, the overall grid security infrastructure should reflect such tiered structure. The aim of this section is to go beyond this statement, showing that one main advantage in considering tiered grids is because this suggests a security approach which allows to overcome some of the drawbacks found in current grid implementations (c. §4).

Following a tiered approach, grid security should be implemented in a modular fashion, such that the security mechanisms satisfy the following requirements:

- for any tier, they must be reliable, efficient, easy to use and appropriate;
- at tier n ($n = 2,3$), they are implemented by reusing (as much as possible) mechanisms for the tier $n-1$ and making them to operate mutually via an extension module for tier n.

The first requirement is the quintessence of the *Principle of Effectiveness* [PFL03]; whereas the second one is a direct consequence of the *Economy of Mechanisms* Principle [SAL74], which itself requires some explanation. Economy of mechanisms means to keep their design and implementation as simple and small as possible. That requirements applies in general in systems development, but deserves emphasis for security mechanisms because of the critical role they play and their exposure to security exploit. Since the goal of an exploit is to find an unusual, unforeseen interaction with a mechanism in order to subvert its normal flow of operations, then it follows that techniques such as line-by-line code inspection and software formal verification must be used. Of course, the simpler and smaller is a mechanism, the easier is to apply those techniques.

The above modular approach to grid security lead us to consider an operating system as a "degenerate" grid of tier 0. Indeed, some mechanisms of the grid security infrastructure have to be implemented at the operating system layer, so it is natural to extend the second requirement in such a way to encompass the case $n = 1$. That in turn rises some interesting questions:

- What is (if any) the correct layering of security services and tools with respect to those four grid tiers?
- Should grid add-on security facilities be implemented only as grid middleware, or it would be better to realize them by extending operating systems security services, too?

Perhaps exceedingly influenced by the aim to not interfere with local security implementations, Globus theorists and designers [FOS98, FOS99] simply leaved unchanged security mechanisms at the operating system layer, putting all in the *middleware* layer. That appears to be not only the strategy currently followed in the design of security services and tools for grid environments, but the general trend followed in realizing modern distributed systems, which has its main motivation in that middleware allows different networks and systems to interact transparently via standard interfaces and protocols. However, operating systems are evolving environments like grids, and today's products often have a modular architecture which provides for open (eventually de-facto)

standard interfaces for security. Moreover, since operating systems are the building blocks of grids, on which grids expressiveness, scalability and functionalities must ultimately rely, it seems unlikely that grid, and operating system technologies will evolve separately. Finally, we believe that right management and access control mechanisms should not be implemented outside of operating systems, whenever possible, because of the following two reasons:

- resource management is performed by operating systems, so implementation of the above mechanisms at this layer guarantees maximum possible control over resources;
- enforcing right management and access control at the application layer results in some code duplication and, as previously detailed, creates a need for trusted application code.

The resulting layering of the overall grid security infrastructure is sketched in Fig.2: a separate software module, which performs tier-specific functions, is provided for each grid tier, and one more module implements suitable extensions to operating system security codes.

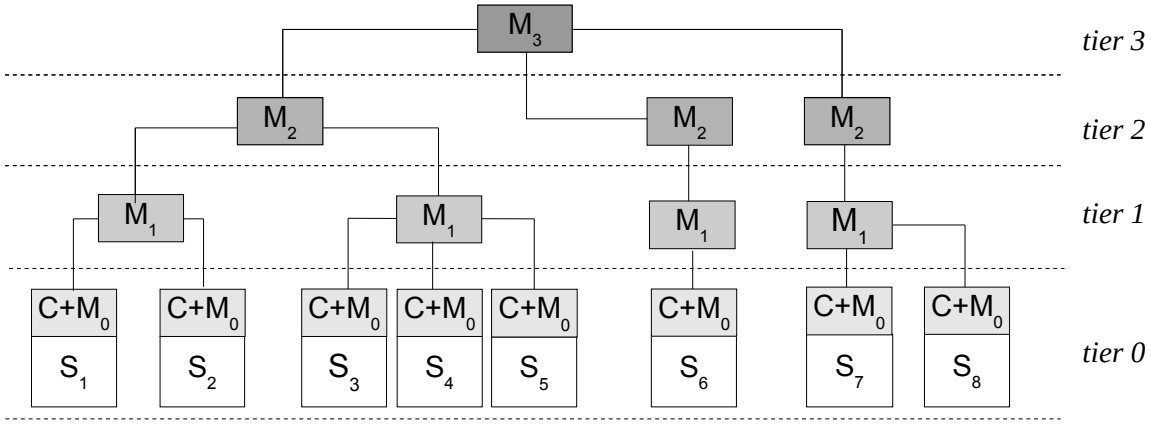

Fig. 2 – *Systems belonging to the same grid cluster* (e.g. S_1 and S_2) *interact by means of: (1) a suitable extension M_0 of general-purpose operating system security code C and (2) a software module M_1 implemented at the cluster grid tier. Interactions in enterprise grid and global grids require their own software modules (denoted as M_2 and M_3, respectively), as depicted above. For example, interactions among S_1* and S_6 *(via a global grid environment) require both M_0,M_1 ,M_2 , and M_3 modules.*

In the following, we outline a possible alternative to the security approach followed in Globus. Our proposal is developed with respect to the model introduced in §3.3, so we discuss basically about design and implementation issues for the T and P spaces, respectively; more precisely, we deal with trustworthiness in §5.1 and with authorization and management of user and process privileges in §5.2. We stress that what follows should be only considered a preliminary work, aimed to suggest possible areas of future research and software development and testing.

5.1.Trust models.

In this subsection we try to determine which are the most suitable models of trust in the T space, depending on the tier of grid considered; and what should be the overall trust architecture for that space as a consequence of the results for each tier and the modular approach depicted previously.

We said in §3.2 that in enterprise and global grids the related trust domains are composite, since they are obtained in a recursive manner from lower level trust domains: cluster-grid trust domains are glued together into enterprise-grid trust domains, which in turn are composed to realize global-grid trust domains. Conversely, cluster-grid trust domains are simple, because the security domain (E,R,π) associated to an operating system doesn't result in a trust domain. If one could relate (E,R,π)

to a trust domain enforced at the operating system layer, then it would be possible to use the same kind of publicly available authentication token (that is, some flavor of a PKC) for authenticating a user directly to any system in a grid environment, and also to authorize a user to perform some action on a system (via a suitable "incarnation" of an AC). More important, asymmetric authentication allows for a direct user-to-user delegation of privileges (c. §5.2).

As we have seen in §3.4, a PKIX in its most basic form consists in a trust domain (E', τ_a), in which a set E' of principals shares a trust relationship τ_a with a single certification authority a. Thus, a natural way to associate a trust domain to a given operating system s seems that of requiring that s acts as a PKIX certification authority for the set U of users registered on s, in such a way that any $u \in U$ (or, eventually, any user belonging to a suitable subset of U, as established by the system administrator) can authenticate to s via his/her public-key certificate. The above authentication proceeds simply as the authentication of a principal to another principal in a PKIX context, the only difference being that now the verifier is the system s (c. §3.4). Of course, that is not intended to be used for an interactive log on, but only during a process-to-process interaction; thus it cannot replace password-based authentication methods. As in the case of password-based methods, a user with multiple accounts on different systems in the same (cluster or enterprise) grid should need just a single private-public key pair. Indeed, that key pair should be related to the identity of an individual in the context of an organization or institution to which he/she belongs. Thus, the previous requirement can be satisfied if:

- s acts as a CA of local user-names;
- s relays on another CA (which we refers therein as *administrative CA*) for the binding of the user public-keys to their identities.

In this way, the service offered by s as a CA results in binding securely user public-keys to user-names, together with their privileges and resources on s. Of course, that requires a chain of trust between s and the administrative CA.

Suppose now that system s gets involved in a cluster grid C. C consists precisely of a set of operating systems and, for each of such systems s, of the set of users, services[12] and resources of s which are enrolled in C. Authentication at the C layer can be implemented via a CA as follows. A system s belongs to C if and only if the cluster CA has issued a public-key certificate for s, as a consequence of a registration phase which is carried out by the system administrator under the responsibility of the cluster administrator. Users can get independently and individually enrolled in C by carrying out a registration phase that results in public-key certificates issued for each of such users u by the cluster CA. The function of the previous public-key certificates is that of binding securely the public-key of a system s or a user u to the public-key of the cluster CA; that is, the assertion "s (or u) belongs to C" is realized via the signature of s (or u) public-key by the cluster CA. Again, all the above requires a chain of trust between the cluster CA and the administrative CA.

Things go about in the same way for enterprise grids. Since an enterprise grid E consists of a set of cluster grids C, authentication at the E layer can be implemented via a CA, whose function is that of realizing the assertion "C belongs to E" via the signature of C public-key. That requires, as usual, a chain of trust between the enterprise CA and the administrative CA.

Since in cluster and enterprise grids there is a single Authority Policy (c. Deff.6, 7), it is natural to associate the administrative CA to it. Eventually, the administrative CA could coincide with the enterprise CA.

Let, finally, consider the case of global grids. Again, global grids G are sets of lower tier (that is, enterprise and cluster) grids; but now, conversely than in the previous two cases, they are characterized by the absence of a root Authority Policy (c. Def.8). Thus, given a global grid G, authentication at the G layer cannot be easily implemented by realizing the assertion "E belongs to G" via the signature of E public-key with the G private key, as it is impractical and unsafe to

12 Sometimes, a system service is impersonated by a fictitious user (e.g. the login user on Unix systems) and in such case we must regard it as an accounting profile (s. below).

collectively manage such private key. In this case, we require that any administrative CA which belongs to the grid has got the list of the public keys of all the grids which compose G, and that the above list is signed by the administrative CA itself. One more again, the different administrative CAs which belong to G must cooperate via a network-of-trust.

We can summarize our approach to trustworthiness in grids as follows. Let (U,S) be a cluster or an enterprise grid. A trust domain for (U,S) can be realized via a single entity a, which results a CA for the set $E=U\cup S$, where U is the set of users and S is the set of operating systems which belong to the grid. The trust relationship τ is such that $p\ \tau\ q$ for any p and q, where p, q may denote both users involved into the grid, processes running on behalf of such users and operating systems in S. Any $s\in S$ has its own certificate (which could be issued by a, as a consequence of a registration phase of s to a, or - alternatively – could be a certificate self signed by s and validated by a via a *reverse-type* certificate [RFC3647]). Any user $u\in U$ has his/her EEC (c. §4.1) issued by a, but he/she has also a certificate relative to his/her account (if any) on an operating system s, which is obtained binding the u public key to the accounting information on s with the s private key.

The above sounds quite different than the authentication approach followed in Globus; indeed, in GSI end entity authentication is provided for users and services, *not* for operating system. Moreover, our approach subordinates users accounts and resources of a system s to its authority. It is important here to point out that users, in this respect, are not only considered as resource consumers, but as resource owners, too: they precisely own resources of s (e.g. files, cpu-time, disk quotas), as defined by their accounting profiles. As such, they can be enabled to delegate a subset of their own resources to other users, thus acting as resource providers, too. That kind of delegation (called *direct delegation*) is at the basis for the enforcement of the *transient collaborations* requirement (c. §2.2), which till today is not supported in Globus, and has been firstly introduced in the prototypical middleware PRIMA [LOR04]. We are going to spend some more word about our proposal for direct delegation in §5.2.

At this point, it may be worthwhile to word how entity authentication takes place in the previous framing, remarking that in our case one has to distinguish not only about user-to-user, user-to-process, and process-to-process authentication, but also about system-to-process and process-to-system authentication[13]. We illustrate those authentication scenarios with respect to the simple global grid environment depicted in Fig.3.

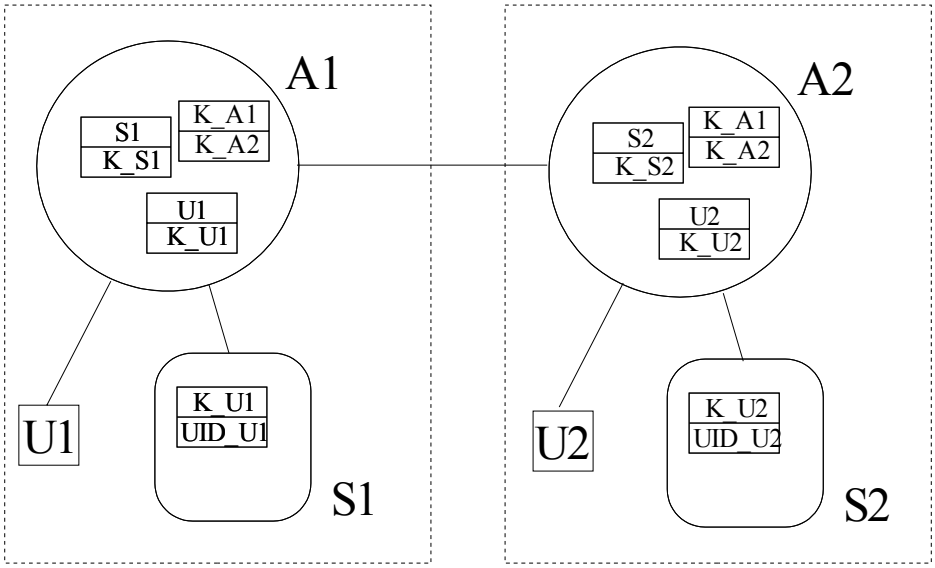

Fig. 3 – *A simple grid environment to illustrate different authentication scenarios.*

13 User-to-system authentication is not considered here, since it is realized by standard mechanisms such as those relying on passwords.

Fig. 3 shows two grids, E1 and E2, which composes a global grid. Each E*i* has its own administrative CA, detoted by A*i*, and enrolls an operating system S*i* and a user U*i*, who has an account on S*i* with UID denoted by UID_U*i*. As a starting point to set up the authentication framing, each entity in E*i* has to generate its own public-private key pair. Then, using suitable integrity-protected channels (c. §3.4), each A*i* can send its authentic public-key K_A*i* to both U*i* and S*i*, and can receive back their authentic public-keys K_U*i* and K_S*i*. After that, all the required certificates can be issued; they are sketched in Fig. 3 as couples such as (U1, K_U1), which denotes a certificate issued by A1 and establishing a secure binding among the identity of user U1 and his/her public key. More generally, a couple (A,B) sketched in entity C indicates a certificate issued by C and having A and B as its main fields. That premised, the authentication scenarios are as follows:

- *user-to-user authentication.* If user U1 wishes to authenticate himself to user U2, then the authentication protocol A1-A3 (c. §3.4) is run with U1 as claimant and U2 as verifier, and (U1, K_U1) has to be checked;
- *user-to-process authentication.* If user U1 wishes to authenticate himself to a system process on S2, then:
  - if the process is impersonated by a user, then it results in a user-to-user authentication;
  - else, the authentication protocol A1-A3 is run with U1 as claimant and S2 as verifier, and (U1, K_U1) has to be checked;
- *process-to-process authentication.* If a process on S1 wishes to authenticate himself to a process on S2, then:
  - If the identifier of process running on S1 is equal to UID_U1, then the authentication protocol A1-A3 is run with S1 as claimant and S2 as verifier, and (K_U1, UID_U1) has to be checked;
  - else, the authentication protocol A1-A3 is run with S1 as claimant and S2 as verifier, and (S1, K_S1) has to be checked;
- *system-to-process authentication.* If S1 wish to authenticate himself to a process on S2, then the authentication protocol A1-A3 is run with S1 as claimant and S2 as verifier, and (S1, K_S1) has to be checked;
- *process-to- system authentication.* If a process on S1 wishes to authenticate himself to S2, then things go in the same way than in process-to-process authentication.

In any case, A1 and A2 are both involved in the protocol as certification authorities, and (K_A1, K_A2) acts as a reverse-type certificate for both of them.
It should be clear from the previous observation that, if a certificate is not an EEC and is issued in the enterprise layer, then it is used to manage trust relationships among CAs in a global grid context. What about non EECs which are related to grid tiers and are considered with respect to a cluster or an enterprise grid? If, as supposed, grid entities refers to a single CA for the enforcement of identities, then the above certificates aren't used at all in the authentication processes. However, they are publicly available and contain integrity-protected information about grid resources; thus they can be used by *resource brokers* to collect reliable information about available resources for a given computation. Moreover, authentication at the different grid tiers can be used to enforce authentication-based routing from one tier to another, thus leveraging overall grid security.

5.2.Authorization and access control.
VOs are created when computational and intellectual resources from separate organizations are shared to tackle problems that require combined efforts and resources; as such, VOs can range from small, ad-hoc groups that may exist for only a short period of time, to larger, well-structured and persistent collaborations.
Recent works in the field of grid technologies (e.g. [LOR04]), indicate that such ad-hoc, transient collaborations strongly depend on the facility of *direct delegation* of user's rights, that is the capability of users to delegate some other entities (users or processes) their rights concerning resource usage in a transparent and automatic fashion, avoiding administrative intervention.

Actually, today's operating systems support for direct delegation of user's rights via access control lists and related mechanisms, but they rely on system-wide authentication schemes which only provide identity delegation. As a consequence, an operating system s can support for rights delegation from an entity a to another entity b, only if both a and b are registered users in s or processes which act on behalf of such users. That, in turn, constitutes a drawback in case of distributed, transient collaborations: if an user a , registered on s , wishes to grant an individual b (which is not a user registered on s) to use the resource of s according to his/her accounting profile, then an account for b must be provided. Direct delegation is especially powerful when combined with fine-grained access rights. The ability to delegate access rights directly is also needed to provide for the necessary scalability of grid security solutions. If delegation is only supported via indirect (third party) means, the overhead required to delegate authorization creates barriers to the scalability of such a system.
The approach to system authentication outlined in the previous subsection can provide for the direct delegation of privileges both in user-to-user and user-to-system interactions. Indeed, using suitable ACs, both an operating system s (through its administrator) and users which have an account on s can delegate an entity e to have some rights on the resources of s. When e submits to s an AC which delegates to it some privileges on s, then s checks the AC with respect to the AC issuer identity (that is, an user account u on s or s itself) and grants rights to e on the basis of the privilege specifications stated in the AC. Since at the operating system level users are identified through UIDs (c. §3.2), if e hasn't an accounting profile on s, then s could supply at runtime an ephemeral UID and a related ephemeral allocation profile (built on the basis of the privilege specifications stated in the AC) to enable (processes acting on behalf of) e to perform actions on the resources controlled by s.

6.Conclusions

In this paper we try to establish if, where and how current grid security technologies fail to meet the requirements imposed for such computing environments, indicating possible alternatives and areas of future research.
We carry out our analysis using an abstract model for grid security architecture which takes into account the Globus Toolkit as a concrete grid implementation of reference, but thinks of global grids as computing environments layered into three logical levels of deployment, depending on their functional and dimensional scale.
Our analysis shows that the main drawbacks of current grid implementations depend basically on the absence of adequate authorization mechanisms for accessing and managing grid resources, and that those mechanisms should be implemented at the operating system layer, because of various security constraints.
Finally, we outlined a tiered approach to grid security which aims to overcome the above drawbacks by extending the PKIX authentication and authorization framings to operating systems, in order that their security mechanisms became fully inter operable with those implemented in the grid middleware. The adoption of PKIX extensions goes towards the use of open standards and protocols, which is an imperative requirement in the design of distributed systems.

References.

[ARK04] T.D. Arkwright, H.J. Schwarz - *Grid-Computing Architecture for Design Automation in Higher Education* - (www.sun.com/products-n-solutions/edu/whitepapers/pdf/gridcomputing_architecture.pdf)

[AVAKI] Sybase Avaki EII Page (www.sybase.com/products/developmentintegration/avakieii)

[BAR02] M. Bartel et al – *XML Signature Syntax and Processing* - W3C Recommendation, 2002

[BOX00] D. Box et al – *Simple Object Access Protocol (SOAP) 1.1* - W3C Note, 2000

[CAN99] R. Canetti et al - Multicast Security: A taxonomy and Some Efficient Constructions – IEEE INFOCOM, 1999

[COND] Condor Project Homepage (www.cs.wisc.edu/condor/)

[COO02] J. Coomer, C. Chaubal - *Introduction to the Cluster Grid- Part1* – Sun Blueprints Series- (www.sun.com/products-n-solutions/edu/whitepapers/pdf/cluster_grid_intro_p1.pdf)

[DEL02] G. Della-Libera et al – *WS Conversation Language*, 2002 (msdn.microsoft.com/webservices/default.aspx?pull=/library/en-us/dnglobspec/html/ws-secureconversation.asp)

[DOD85] U.S. Department of Defense - *Trusted Computer Systems Evaluation Criteria* – DOD5200.28-STD, 1985

[EAS02] D. Eastlake, J. Reagle, Editors – XML Encryption Syntax and Processing – W3C Recommendation, 2002

[EGEE] EGEE Project Homepage (public.eu-egee.org/intro/)

[FOS98] I. Foster et al - A Security Infrastructure for Computational Grids - *Proc. 5th ACM Conference on Computer and Communications Security Conference*, 1998

[FOS99] I. Foster, C. Kesselman. (eds.) - *The Grid: Blueprint for a New Computing Infrastructure* - Morgan Kaufmann, 1999

[FOS01] I. Foster, C. Kesselman, S. Tuecke - The Anatomy of the Grid. Enabling Scalable Virtual Organizations - *J. Supercomputer Appl.* (2001)

[FOS02] I. Foster et al - *The Physiology of the Grid: An Open Grid Service Architecture for Distributed Systems Integration* – (www.globus.org/research/papers/ogsa.pdf)

[FOX01] G. Fox, D. Gannon – Computational Grids – *Computing in Science & Engineering*, 2001

[LEGIO] The Legion Project Homepage (www.cs.virginia.edu/~legion)

[LOR04] M. Lorch – *PRIMA, Privilege Management and Authorization in Grid Computing Environment*, PhD. Dissertation, Virginia Polytechnic Institute and State University, 2004

[GEN01] W. Gentzsch- *Grid Computing: A New Technology for the Advanced Web* (www.sun.com/products-n-solutions/edu/whitepapers/whitepaper_gridcomp.html)

[GLOB] – Globus Project Homepage (www.globus.org)

[GLOB02]The Globus Project™- *Introduction to Grid Computing*, PDF presentation 2002 (www.globus.org/)

[GRI99] A. Grimshaw et al - Legion: An Operating System for Wide-Area Computing, *IEEE Computer*, 32:5, 1999

[GT4SD] Globus Toolkit 4 Security Documentation (www.globus.org\toolkit\docs\4.0\security\index.html)

[KOH78] L. M. Kohnfelder – *Toward a Practical Public-Key Cryptosystem* – B. Sc. Thesis, MIT Department of Electrical Engineering, 1978

[KOHL94] J. Kohl et al, The Evolution of the Kerberos Authentication Service, Distributed Open Systems (Brazier F. , Johansen D. Eds.), IEEE Computer Society Press, 1994

[MEN97] Menezes et al, *Handbook of Applied Cryptography*, CRC Press, 1997

[MOAB] Moab Grid Suite Homepage (www.clusterresources.com/products/moabgridsuite.shtml)

[NINF] Ninf Project Homepage (ninf.apgrid.welcome.shtml)

[NPACI] Archives of the National Partnership for advanced Computational Infrastructure (www.npaci.edu/)

[OASIS] Organization for the Advancement of Structured Information Standards Consortium (www.oasis-open.org/home/index.php)

[ORAC] Oracle Real Application Clusters Homepage (www.oracle.com/database/rac_home.html)

[PFL03] C.P. Pfleeger, S.L. Pfleeger - *Security in Computing*-Prentice Hall, 2003

[RFC2246] T.Dierks, C. allen, *The TLS Protocol Version 1.0*, IETF Network Working Group, 1999
[RFC2692] C. Ellison, *RFC2246: SPKI Requirements*, IETF Network Working Group, 1999
[RFC2693] C. Ellison et al, *RFC2693: SPKI Certificate Theory*, IETF Network Working Group, 1999
[RFC2743] J. Linn - *Generic Security Services Application Program Interface Version 2, Update 1* - IETF Network Working Group, 2000
[RFC2744] J. Wray - *Generic Security Services API: C-bindings* - ETF Network Working Group, 2000
[RFC3647] S. Chokhani et al, *RFC3647: Internet X.509 Public Key Infrastructure. Certificate Policy and Certification Practice Framework*, IETF Network Working Group, 2003
[RFC3820] S. Tuecke et al, *RFC3820:Internet X.509 Public-Key Infrastructure (PKI) Proxy Certificate Profile,* IETF Network Working Group, 2004
[SAL74] J.H. Saltzer, M.D. Schroeder – The Protection of Information in Computer Systems, *Communications of the ACM* 17, 7 ,1974.
[SUN1] Sun N1 Grid Engine 6 Homepage (www.sun.com/software/gridware/index.xml)
[SWA99] V. Swarup, J. T. Fàbrega – Trust: Benefit, Models and Mechanisms – *Secure Internet Programming- LNCS State-of-the-Art Survey*, Springer, 1999
[UNIC] UNICORE Project Page (unicore.sourceforge.net/unicoreatsf.html)
[W3CWS] World Wide Web Consortium - Web Services Activity home: (www.w3.org/2002/ws/)
[WSSC1.1] *Web Services Secure Conversation Language Specifications , version 1.1* (www6.software.ibm.com/software/developer/library/ws-secureconversation.pdf)
[X509v1] Information Technology – Open System Interconnection – The Directory – Authentication Framework ISO-IEC STANDARD 9594: 1990-8 | CCITT X.509, 1989
[X509v3] Information Technology – Open System Interconnection – The Directory – Authentication Framework ISO-IEC STANDARD 9594: 1993-8 | ITU-T X.509, 1993
[ZIM95] P. Zimmerman, *The Official PGP User's Guide*, MIT Press, 1995