Security of Distributed Data Management

Frohner, A
(EGEE, JRA1, DM) *et al*

18 April 2006

EGEE is a project funded by the European Commission
Contract number INFSO-RI-508833

The electronic version of this EGEE Technical Reports is available
on the CERN Document Server at the following URL:
<http://cdsweb.cern.ch/search.py?p=EGEE-TR-2006-003>

EGEE

Security of Distributed Data Management

Document identifier:	**EGEE-TR-2006-DataSec**
Date:	**April 11, 2006**
Activity:	**JRA1: Data Management, CERN**
Document status:	**DRAFT**
Document link:	**https://edms.cern.ch/...**

Abstract: Security is a pervasive concept that cannot be decoupled from any distributed systems design. The constraints imposed by security considerations have a profound impact on the overall design of the services and interfaces of any Grid middleware and infrastructure, especially for data management. In this article we discuss the requirements, architecture and design of the security mechanisms that can be put in place for distributed data management. The advantages and disadvantages of different solutions and deployment models are discussed. We illustrate the discussion using the implementation of the EU-funded Enabling Grids for E-SciencE (EGEE) project. We direct our focus on the security aspects of data access, data replication, data location and data consistency and how the security requirements of the applications are being addressed in the EGEE middleware stack *gLite*.

1. INTRODUCTION

In distributed systems design, security cannot be orthogonalized as a concept. Securing the system cannot be done separately, and this has a profound impact on the actual architecture and design. In Grid infrastructures as we define them, there are many constraints due to security considerations.

We define the **Grid** as systems and applications that aim to integrate, virtualize, and manage resources and services within distributed, heterogeneous, dynamic Virtual Organizations across traditional administrative and organizational domains [5, 10]. A Virtual Organization (VO) is composed of individuals and/or institutions having direct access to resources (computers, software, data, network, etc) that they intend to pool for collaborative purposes. The concept of Virtual Organizations is what distinguishes the Grid as a concept from other known domains of computing (such as distributed computing, cluster computing). For the security discussion it is essential to understand the trust relationships and boundaries that are set between the resource owners and the Virtual Organizations [2]. In the section on requirements we will go into the details of what is required of these relationships, keeping the focus on data management.

The key to be able to define and operate a secure Grid infrastructure as we will illustrate it in the EU-funded Enabling Grids for E-sciencE project (EGEE), is to build on standards wherever available, or to propose standardization wherever necessary. The paradigm that is followed by current middleware stacks (like the Globus GT4 and the EGEE `gLite`) is called *Service Oriented Architecture*, which aims to facilitate interoperability among Grid services and to allow easier compliance with emerging standards such as OGSA [10]. The security framework chosen by most Grid middleware stacks, including GT4 and `gLite`, relies on the Grid Security Infrastructure GSI [15, 18].

In GSI, the X.509 infrastructure identifies the Grid user via a certificate, which is used for authentication of the user when it contacts the Grid services. Each user acquires the certificate from a Certificate Authority (CA). The resource owners trust the CA, and thus each user that the given CA has certified. The middleware often needs to act on the users behalf, so in many cases the user has to **delegate** its certificate to the service. Then the service will contact other services in the user's name, as if the user had contacted these services himself. In order to avoid abuse and infinite propagation of delegated certificates, the Grid certificates are only short-lived proxies of the user's certificate. Proxies expire after a short amount of time (usually 12 hours). For Grid jobs that need an extended amount of time to complete, the users have to extend and renew the proxies so that such jobs can also be supported properly.

In terms of services, there may be services serving only one VO (VO-owned services) and services serving many VOs. There are also site-services that are independent of any VO and may serve any VO or just selected VOs depending on the site resource owner's agreement with the VOs. So although many services are managed by a VO, there is no requirement of having independent service instances per VO. For performance and scalability reasons service instances will in most cases serve multiple VOs.

Security services encompass the Authentication, Authorization, and Auditing services which enable the identification of entities (users, systems, and services), allow or deny access to services and resources, and provide information for post-postmortem analysis of security related events. They may also provide functionality for data confidentiality and a dynamic connectivity service, i.e. a means for a site to control network access patterns of applications and Grid services using its resources. In terms of data management and data access, security semantics are often very difficult to synchronize in a heterogeneous and distributed computing infrastructure. In addition, different Virtual Organizations have sometimes different and even conflicting requirements concerning security, or their requirements are posing serious problems to the resource owners. Therefore it must be possible both for the VOs and the Grid sites to choose their trust domains, i.e. with whom they intend to collaborate and with whom they cannot do so.

In this article we will illustrate the security concepts through the EGEE middleware stack `gLite`. We explain and introduce the services in the first half of the paper. The most notable security service that is introduced in `gLite` is the VO Membership Service VOMS [12]. VOMS allows an authenticated and authorized administrator to manage the VOMS database, and an authenticated user (or any principal) to

request membership of a VO, and request group membership, role, and capability entitlements. Once the user has been granted the appropriate VO membership and attributes within a VO, he may request a short lived credential. The user runs `voms-proxy-init` with optional arguments relating to which VOs, groups, roles, and capabilities he wishes for his current credential. VOMS issues a short lived Attribute Certificate [30] to the authenticated user, which the user may then present to resources on the Grid. The credential is in the form of a private, non-critical extension included into the user's X.509 proxy certificate. The inclusion allows transparent transfer of the credential, allowing basic GSI services to use the proxy as before. VOMS allows use of a push model, the service does not have to pull VO membership of all potential users. To provide backward compatibility VOMS can also fall back to a pull model if need be to serve membership information for the grid-mapfile generation. VOMS supports membership of multiple VOs, hence a VOMS proxy certificate may contain information from more than one VO.

For the time being, `gLite` supports only the transport level security infrastructure as opposed to message level security (see [16] for a discussion on differences). The transport level security builds on TLS [11] for authentication using X.509 proxy certificates.

In the rest of this article we will give a concise summary of the security requirements and boundary conditions for data management in a distributed Grid infrastructure. We will illustrate how `gLite` [5, 6] addresses these requirements in EGEE, which is the largest international research Grid computing infrastructure today, focusing on the specific issues around secure storage and access of data.

2. Data Security Requirements

The concept of Virtual Organizations supplies a context for operations on the Grid that can be used to associate users, their requests, and a set of resources. The sharing of resources in a VO among the VO users has to be tightly controlled, resource providers and consumers being able to define clearly the extent of what is being shared, who is allowed to access and to share resources, and the conditions under which sharing occurs. This resource sharing is facilitated and controlled by the Grid Middleware that is composed of a set of services providing a layer between physical resources and applications.

We try to categorize the requirements on the Grid Middleware in terms of securing data on the Grid from the point of view of the entity formulating the requirement. This list of requirements is specialized and focused on the data management aspects. See a good overview of general grid security requirements in [2] and [4].

2.1. User Requirements

The average Grid user may have very limited or very strong requirements on data security.

[U1 No Security] There are applications and users that really don't care about security at all. For this category of users, performance is the most important aspect and security of data access always comes with a performance penalty. The user requesting no or limited security has no issues with others being able to read his data. Upon insisting though, even these users will admit that it would not be in their interest to leave their data open for everyone to delete as other people may simply do so when they need the storage space for themselves.

[U2 VO Access Only] A reasonable variation of the *no security* scheme is where the VO controls the data, i.e. everyone who is member of the VO has the same rights on all the data. Everyone in the same VO can read and write everything that is 'VO data'. Most VOs require a distinction between read and write operations though, which boils down to defining two groups only - readers and writers. Some High Energy Physics VOs request such semantics.

[U3 Unix semantics for Files] The previous two cases apply to data stored either in files or in databases. Unix semantics of course only make sense for Files. For most VOs this is the expected semantics. The owners and groups are to be taken using the user's certificate DN and the groups in the VOMS attribute certificate.

[U4 Posix ACLs for Files] This is a well-understood extension on the previous unix semantics by adding access control lists to each file.

[U5 Additional Operations in ACLs] Additional information on the files or new operation semantics may require additional operation entries over the basic (read, write, execute) permissions, such as *delete* (on a file), *set/get-metadata* and *replicate*.

An indirect requirement we can distill for the Grid is that the middleware needs to be **flexible**enough to support the semantics that the users of a given VO want to have **[U6]**. It is commonly accepted that the semantics are defined on the VO level for all users in a given VO, i.e. there is no use case for VOs where some members require no security while others need ACLs.

The users also require the security on data to work **uniformly**. Most data is replicated across many sites and is available from more than one data store. It must not happen that the access control is different from one store to the other. A user expects to be able to access the same data everywhere and does not want to have to care whether he can access data at one site but not at another one **[U7]**. This mandates the synchronization of the access control information across the data stores in the Grid.

Another requirement from the user's point of view is about the ability to **access existing data** at the user's local site as before, without the Grid. The requirement is about the transparent but secure access of data using local access control and Grid access control as well **[U8]**. This can be nontrivial as local access mechanisms may have different semantics from Grid access and users may have different sharing semantics, making it nontrivial to fulfill this and the previous requirement at the same time.

2.2. VO REQUIREMENTS

The Virtual Organizations have also different security needs depending on the user community that the VO serves. The VO is focusing on different kinds of security requirements as the users, who are only concerned with their own data security and being able to access data necessary for their Grid jobs to complete.

[V1 Storage Space] A VO needs to secure a certain amount of storage space for its users, and it has to make sure they do not over commit the allocated space (quota). The VO has to be given the security that the space that it may have to pay for is indeed available and usable for its users at the Grid sites that the VO has service level agreements with.

[V2 User Management] The VO needs to be able to manage its users autonomously and independently of the Grid sites. The users need to be able to have groups or roles assigned to them, which form the basis for the authorization decisions concerning data access in the Grid.

[V3 Encryption] Some VOs may have to encrypt their data in order to be able to distribute it outside of trusted sites. Encryption and decryption facilities with the corresponding key management needs to be available from the Grid middleware to be used by the VO members.

[V4 Transfer Management] The VO may need the ability to control and secure some of the bandwidth available for transfer between sites, at least for a certain amount of time. Often VOs need to distribute data, e.g. there may be a data source providing vast amounts of data that need to be distributed across the many Grid storage nodes in order to be analyzed later. The VO may also want to be able to order the queue among its users.

[V5 Site Selection] The VO may not trust a set of sites and needs to be able to exclude explicitly some sites from the list where its data may be stored.

[V6 Accounting] The VO may request the possibility to account for the usage of space in detail if it wants to control how the VO members are make use of the space.

Also the VO is keen on **ease of use and ease of configuration** for the security options that it has available on the Grid infrastructure **[V7]**. As mentioned above, it is very likely that some VOs will have more requirements on security than others, and the infrastructure has to be able to support as many security models as possible in parallel for different VOs **[V8]**.

2.3. Resource Owner and Site Requirements

Sites administrators and resource owners are more concerned with accounting, auditing and the enforcement of their local rules and regulations.

[S1 Transparent User Management] The site would like to manage the Grid users on the VO granularity, as it is not practical for every Grid user to have to sign up at each Grid site. Sites and VOs have to establish trust relationships and reach maybe signed Service Level Agreements so that the administrative boundaries and trust domains are well defined and enforced.

[S2 Site Control] Sites usually trust the VOs to do their individual user administration and expect the VO to manage their members and the user rights through VOMS as explained in the previous requirement. However, sites usually need to have a 'last word' in who is actually able to use their resources and expect e.g. a possibility to ban individual users or even complete VOs. The sites must be in control of their own resources.

[S3 Accounting] Sites have to be able to account for their storage usage, usually in the VO granularity.

[S4 Auditing] Auditing of use upon request is prescribed by law on some sites. This means that a detailed audit trail for data access has to be kept in some places and the Grid middleware has to allow and provision for that.

Ease of management and ease of configuration is probably the most important requirement for the sites **[S5]**, as the site administrators usually have no interest or possibility to spend additional time on Grid middleware configuration.

2.4. Generic Grid Security Requirements

Due to the heterogeneous nature of the Grid, and in order to fulfill the requirements stated above, we have to take into account a few additional requirements. These have been formulated already in other places like [2]. For EGEE we had to make some choices in order to fulfill some of these requirements, and these choices may have an impact on the interoperability of `gLite` with other Grid middleware.

[G1 Heterogeneous Security Infrastructures] In a generic Grid setting, the data may move through centers with different infrastructures, e.g. some might use GSI while others use Kerberos [23]. For EGEE we have circumvented this problem by requiring each participating to use GSI for the Grid. For some sites that use especially Kerberos, there are bridges in place that can translate an incoming X.509 certificate into a local Kerberos token [25] and back to X.509 certificate [24]. This is an arguably strong restriction, which shows that bridging security domains across technology boundaries is particularly difficult.

[G2 Trust Between Sites] It is also necessary to establish trust relationships between centers as it may happen that certain services managed by one site will directly interact with services managed at another site. If there are established agreements between sites, it is also easier for VOs to reach SLAs with the Grid infrastructure as such, as it is also not practical for every new VO to have to sign agreements individually with each site (there are around 150 participating sites in EGEE). In EGEE there are written agreements between the participating sites and the EGEE project which then acts as controller and enforcer of the agreements. This is a nontrivial task and takes up a considerable amount of the resources of the EGEE project.

3. SECURITY MODELS FOR DATA MANAGEMENT

Having the requirements in mind, we can now discuss security models and their advantages and disadvantages. We also discuss how the models address or fail to address the requirements.

The models are distinguished by the layering of the Policy Decision Points (PDP) and the Policy Enforcement Points (PEP). In addition, it matters whether the service that provides the PDP or PEP is a single central instance or whether it is also distributed and has to be synchronized across its instances.

The layers that we consider are the usual Grid Layers as described in [1]. The top-layer is the Application Layer. The bottom layer is the Resource Layer and in-between we have the Middleware Layer. The middleware may be composed of several layers itself, for this discussion it is enough to have this simple three-layer granularity. The five models we discuss are illustrated in Figure 1. The 6th model, where

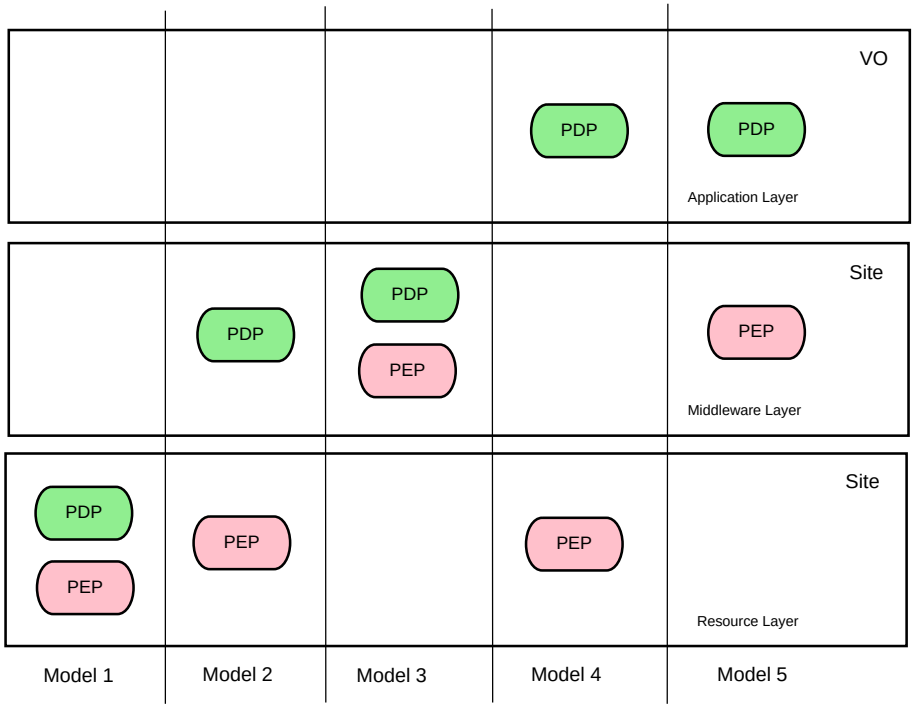

Figure 1: The different security models for data management. The difference is only in which layer we put the policy decision and enforcement points. However, these have profound implications for the trust relationships between VOs and sites, and the implementation. The application layer services are all VO services, whereas for this discussion we restrict ourselves to site-only services in the middleware and resource layers.

both the PDP and PEP are in the application layer is against requirement S2 (site control) and therefore is not up for discussion.

For data management, the PDP is the service that stores the authorization information for the Grid data, and the PEP is the actual data access service that will enforce the decision of the PDP. Of course there needs to be a strong trust relationship between the PEP and the PDP, i.e. the enforcement has to trust the decision maker. For file-based data, it also depends on which level the PDP is in terms of data identification and data naming. There may be several levels of mapping necessary to map the name the application has given to the file to the actual name that is used to access the data in the data store. Depending on which name is used for authorization enforcement, the mappings themselves have to be protected. We will give an illustration on how this is done in `gLite` in the next section.

In the discussion points below we denote:

o neutral comments on the given model

+ advantages

- disadvantages

3.1. MODEL 1: RESOURCE LAYER PDP AND PEP

In this first model, both the PDP and the PEP are on the resource layer. We can assume that in this model the PDP and PEP are integrated into the very same service, like a large hierarchical storage system or a database. In this model the actual data storage available on the Grid sites is responsible for deciding and enforcing the authorization of the user. In order to fulfill the requirements, the resource has to be able to accept and interpret Grid certificates (with the additional VOMS information) locally. Each site service has to be able to do so autonomously, but also in a uniform way (U7).

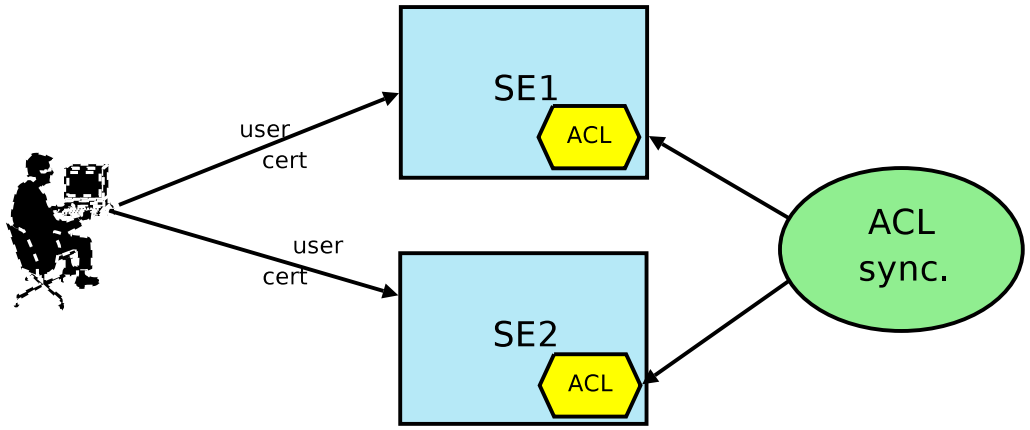

Figure 2: Resource (Storage Element) layer policy decision and enforcement point

- The storage resource (file storage or generic data storage) at the site has to offer rich security semantics in order to fulfill U1-U6. The 'least common denominator' in this case is generic ACLs. So data stores offering generic POSIX ACL support may operate only in model 1.
  - **+** Rich security semantics, such systems also usually offer quota management, making it easy to support V1.
  - **+** Resource controls all security locally (U8 is easy to fulfill)
  - **+** All site requirements are fulfilled as the site is in complete control of its own resource
  - **-** A security interface is not completely standardized yet.

- ACLs are not implemented by many storage systems. If they are, they tend to be proprietary, working on local users and groups, thus one needs an additional mapping step from the grid credentials.

- The storage namespace is relevant - it has to hold the authorization semantics.
  - o For file-based data, the data storage directory names will have to be known and managed explicitly by the users.
  - \+ The namespace has semantics which can be to the advantage of the user.
  - - Standards about file namespace management are only emerging and implementations are still in their early stage.
  - - If users/groups are changed or deleted, each storage resource has to be updated individually for all data owned by the given principal, which is potentially an expensive operation.

- Writing data into file storage: The user has to write into a namespace where it's authorized to write to.
  - o On file creation the directory has to exist and be writable by the user.
  - o VOs have to be given new namespaces on allocation as an administrative operation.

- Reading data from data storage: The user's credentials are checked for authorization (ACLs) locally.
  - \+ doesn't matter what protocol is used to access the data. The authorization decision will be made locally.

- Consistency of PDP security information across sites has to be assured, i.e. the PDP's have to be synchronized across all sites. If data is replicated from one site to the other, all of the security information has to be propagated.
  - o The namespaces of each storage system across the grid have to be synchronized, this follows directly from the points above.
  - o If the data is cataloged in the middleware or application layer, those data catalogs have to match the naming scheme and associated security semantics of the storage name spaces. Or the mapping may also have no security connotations at all, leaving the security and enforcement thereof to the resource layer. In that case however, the catalogs in the middleware and application layers cannot be used to check whether the file is accessible by a given user.
  - - A dedicated consistency service will have to be provided that makes sure that replicas across data stores have the same authorization information.
  - - Renaming files and changing ACLs will become very expensive distributed operations.

This model enables the site to exert full control over the data that is stored in it. Auditing and Accounting are relatively straightforward tasks and performance of access is as good as the resource is able to provide, there are no additional services to be called.

Although this model has a lot of advantages and is very attractive to the site administrators, the negative points are substantial. Storage systems may support POSIX ACLs, NTFS-like ACLs, etc if they support ACLs at all. Put together with the point about synchronization, it illustrates that distributed security infrastructures rely heavily on available and implemented standards to work, and as long as they are not available, even the simplest model is difficult to deploy. For file-based data, the GGF Grid Storage Management Working Group is standardizing on the Storage Resource Management interface (SRM)

[9]. SRM v1.1 has no means of accessing ACLs, SRM v2 has limited support (semantics are not fully defined) and SRM v3 is not through standardization yet.

In addition, the naming issue cannot be completely neglected as for many security models (file-based ACLs) the semantics are tied to the hierarchical namespace. For example, a whole directory tree may have to be created at the target storage to replicate a file. This is problematic as the ownership of higher level directories may be different from the actual file, in addition the principal replicating the data may not be the owner of the file at all. So the service performing the replication operation has to have additional privileges that the user does not have and needs to be able to communicate these to the target storage service so that the security information can be propagated at all.

3.2. MODEL 2: MIDDLEWARE LAYER PDP, RESOURCE LAYER PEP

In this model, the policy decision point is provided by the middleware and not the resource. This means that there is a strong trust relationship between the resource and the middleware component that stores the authorization information (File Authorization Service – FAS). The model assumes that the middleware PDP is controlled by the same site as the PEP (the data storage) so this trust relationship should be straightforward to establish as it will be usually the very same site administrator responsible for both services.

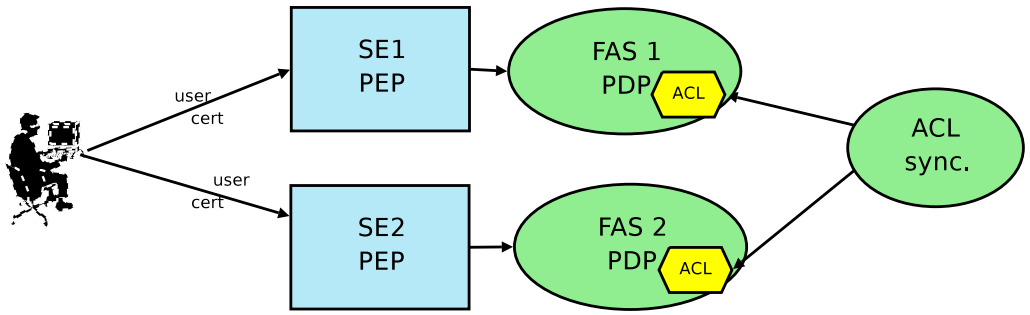

Figure 3: Resource (Storage Element) layer policy enforcement point; Middleware (File Authorization Service) layer policy decision point

So this model is very similar to model 1, except that the data catalog storing the authorization information (i.e. the ACLs) is not part of the data storage resource but it is a Middleware service. Still it is very tightly coupled to the storage itself. This model may be chosen if the storage resource does not natively implement a hierarchical namespace for files for example. Then the middleware can supply this functionality which together with a tight coupling results in the same system as the one in model 1, with the same advantages and disadvantages. The additions are listed below.

- The authorization information is a stored in a middleware catalog.
  - **o** The data resource needs to be able to perform an authorization call-out to adhere to the PDP decision. This usually mandates the implementation of the call-out at the resource. This authorization call-out has to be called for every access mode of the storage resource.
  - **+** The semantics of the authorization information is easier to standardize on as it is now not up to the resource providers but to the middleware to agree on the semantics.

3.3. MODEL 3: MIDDLEWARE LAYER PDP AND PEP

In the model where the middleware layer is in control, a Grid access service does the enforcement of the data authorization and access control. This means that all data access has to go through the Grid

middleware service 'door', the direct 'backdoor' access to the resource layer will not be authorized.

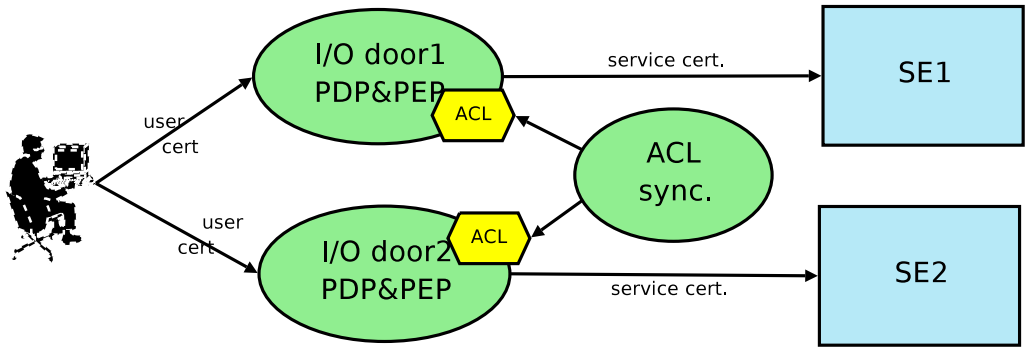

Figure 4: Middleware (I/O door) layer policy decision and enforcement point

If the middleware layer has both the PDP and PEP, the security semantics can be completely decoupled from the underlying data storage implementation. The middleware layer will 'own' the data in the storage and all grid access has to go through the middleware layer where it will be properly authorized. The middleware can then enforce any kind of semantics on top of the storage, even richer ones that the data store on the current resource can actually provide, if at least it can guarantee that no other access mode is possible (no backdoor access).

In this model we still require the middleware services to be site services.

- ACLs are stored in the Grid middleware.
  - **o** The data and/or file names in a resource data store do not need to carry any semantic information in their namespace, tied to authorization. From the security point of view, these names are opaque. We usually call these names Storage URLs (SURL).
  - **o** There will need to be another middleware-layer namespace that carries now this information. We usually refer to this namespace as the Logical namespace, where file-like data has Logical File Names. The mapping of LFNs to SURLs is kept in the middleware catalogs and is secured according to the necessary semantics. The SURL does not have to be in sync with the LFN namespace, which means that orphaned files cannot be associated with anyone based on just the SURL. For all accounting purposes the grid access door has to be used and it has to provide the site with proper accounting and audit logs.
  - **o** 'Rename' and 'change ACL' operations are a little cheaper as the uniformity of the semantics can be guaranteed by the middleware. The distribution of the LFN catalog and its synchronization across sites is still an issue, just like in the previous two cases.
  - **+** ACL synchronization across sites does not depend on the local data storage semantics.
  - **-** The File Authorization Service has to be distributed and synchronized separately.
- Writing into the storage has to go through the grid access door. The backdoor (direct native I/O to the storage device) has to be secured (i.e. the grid service owns all the data in the storage resource).
  - **-** each storage device has to be interfaced with the Grid access service. A Grid I/O service has to be provided.
  - **+/-** Replication including all security data has to be done via a dedicated middleware replication service. After the data transfer it has to update the local PDP information with the proper full ACLs. This again is a 'superuser' operation which should only be allowed to be done by the dedicated replication service. So here we end up with a network of trusted grid services

that are allowed to do 'superuser' operations on the Grid storage access doors. With X509 this can get complicated as both the service and the user credentials have to be passed and evaluated at the PDP at connection time.

- Reading data from the storage has to be authorized by the PDP.
  - The data authorization can be potentially complicated and expensive operation in terms of performance.
  + It's consistent across different storage implementations and does not need any synchronization of additional storage resource metadata.
- Synchronization of data across sites and between access methods (backdoor access).
  - In this model data that has been put into the storage through other means than the grid access door has to be 'moved' to the grid authorization domain explicitly - this may be an administrative operation.
  - Back-door changes cannot be tracked and will destroy consistency if used carelessly. Checking for backdoor changes is expensive and usually not practical.
  + Consistency across sites is as good as the Middleware PDP distribution allows it. Distributing the authorization catalog is probably an easier problem than to agreeing with all data resource implementations on the authorization semantics (model 1).
  + As mentioned already, also data resources with very limited security capabilities (e.g Unix) can be taken as a fully secure Grid storage service using ACLs.

This model allows to interoperate with sites that have any kind of data storage resource, not burdening the sites to have to implement or provide storage authorization semantics matching the rest of the Grid. However, the synchronization problems are the same and the scalability of the Grid access door will become an issue if the implementation is not good enough.

3.4. MODELS 4 AND 5: APPLICATION LAYER PDP

In Model 4 and 5 the PDP is a VO service and is outside of the site's control. Model 4 places the PEP on the resource layer and model 5 in the middleware layer, but the differences here are the same as between model 2 and model 3, so we focus only on the consequences of putting the PDP on the application layer. If the site is to trust the PDP's authorization decision, the PDP has to be authenticated towards the site as well (i.e. the site PEP has to be sure that the PDP that it asks for authorization information is indeed the PDP it claims to be. Inter-service authentication is not provided by the Grid middleware yet.

However, this model has the advantage that the PDP in the application layer may be a single central instance, so the synchronization of authorization information between sites may not be an issue anymore.

- The PDP is VO-managed and may be a single central instance.
  + Rename and change ACL operations are cheap.
  + Synchronization across sites is not an issue.
  - Some sites may simply refuse to accept decisions on authorization made outside of their security domain.
  - Scalability is limited and it is a single point of failure. If the VO PDP service is unreachable, no data can be accessed anywhere.
- The PDP may be distributed only across a few sites.

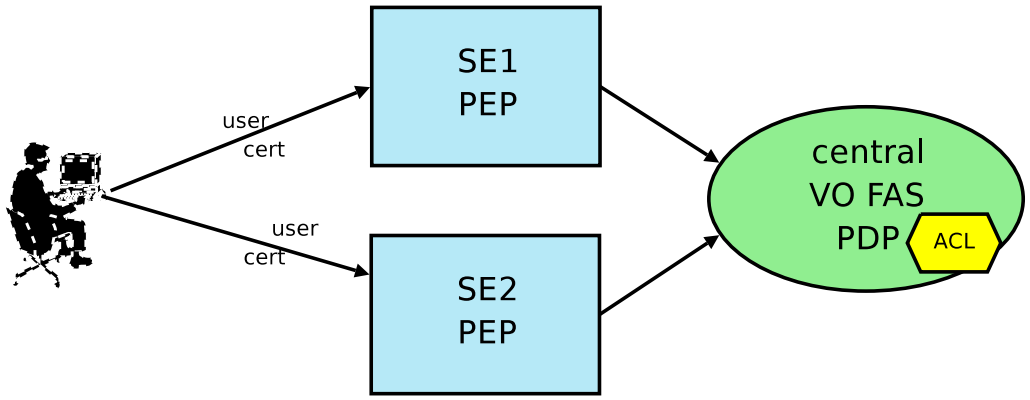

Figure 5: Resource (Storage Element) layer policy enforcement point; Application (central VO File Authorization Service) layer policy decision point

+ Synchronization may be solved by simple replication techniques (like database replication),
+ The service is not a single point of failure anymore.
- Performance is still a big problem and the trust relationship to the sites.

3.5. PEP – PDP Interaction Types

There are three basic interaction types that can be used to communicate a decision to the policy enforcement point:

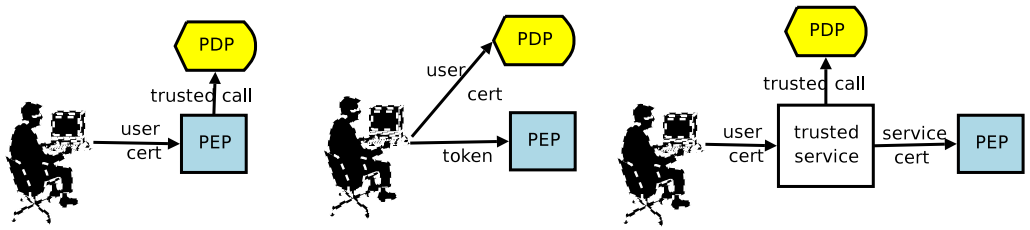

Figure 6: PEP — PDP Interaction types: (i) pull (ii) push (iii) trusted

pull A call-out from the PEP to the PDP for authorizing the given user's credentials for the operation.

push Accept a token that has been signed by the PDP that the user is indeed allowed to perform the requested operation (CAS model) [26].

trusted Trust the service that is trying to read the data on the user's behalf as a superuser service.

These interaction types could be combined with any of the models above.

4. Data Services in EGEE

In this section we are describing the services that are available in EGEE for data management and show how the models discussed before are applicable. In principle, as explained above, the models can coexist

and interoperate. We describe how far we got with the implementation of a system that can accommodate the different models and where are the caveats and where are the points where still work has to be done.

The gLite implementation of the data services has three basic components: The Storage Element, the File Transfer Service and the Data Catalogs (file and replica, metadata and encryption key store catalogs). The catalogs also provide database access control in this context. These components are described in this section in some detail, in order to be able to discuss how the different security models can be accommodated.

4.1. Storage Element

In gLite we define a Storage Element (SE) as having at least three interfaces (see Figure 7). The Storage Resource Management (SRM) interface [9] allows the client to manage the storage space - to allocate space for jobs, to prepare data to be retrieved through a certain protocol, etc. The second interface that is expected to be available on a SE is a native POSIX-like file I/O interface. And finally, the SE is expected to provide a transfer protocol interface, where we expect at least GridFTP [17] to be supported.

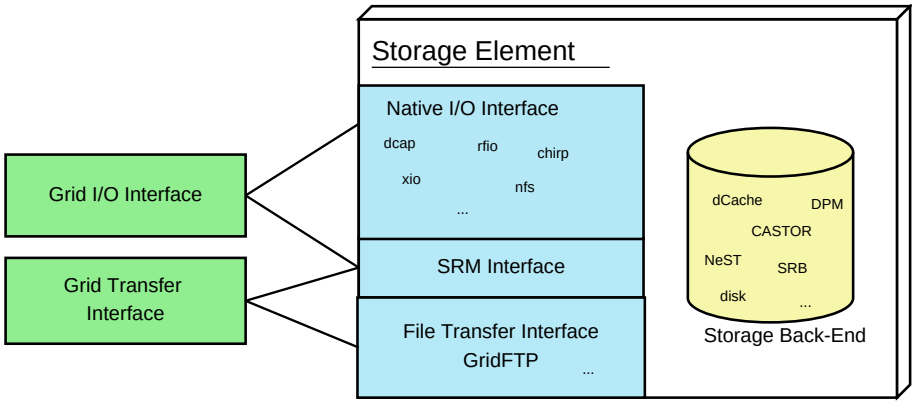

Figure 7: The Storage Element. The SE has three external interfaces: GridFTP, SRM and a native POSIX-like file I/O interface. Depending on the underlying storage system, the SE may support more than one protocol to access the files. The list of examples for storage and I/O in the picture is non-exhaustive. For EGEE we expect each storage to support at least GridFTP as a common transfer protocol, of course also other protocols may be supported. The two interfaces on the left are the Grid interfaces to storage: a Grid I/O and a Grid Transfer interface.

The SE Grid interface implementations will interact with the available storage and protocol components. The SE exposes a uniform interface to the Grid clients and abstracts away local peculiarities (like the locally available native I/O protocol). The two Grid interfaces are a Grid I/O and a Grid Transfer interface. In EGEE these are provided by the gLite I/O and the File Transfer Service (FTS).

4.2. File Transfer Service

The data movement services provide scalable and robust managed data transfer between Grid sites, to and from Grid storage. Files are scheduled to be moved between sites reliably. Scalability, manageability and extensibility have driven the service decomposition of the gLite data movement services.

The logical service decomposition for data services is shown in Figure 8. The services involved in data movement are the following:

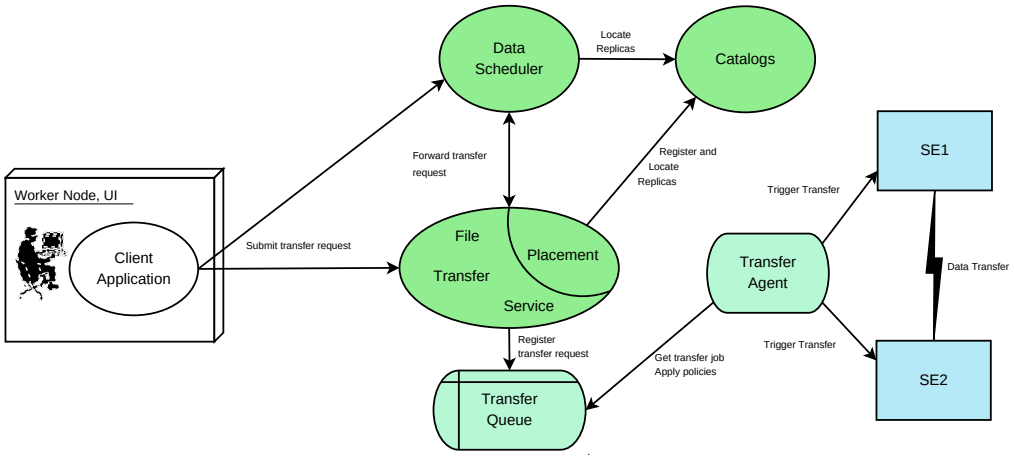

Figure 8: Architecture overview of the Data Movement service components. The Data Scheduler is a high-level component. There may be many File Placement Services around, acting as interfaces to put requests into the Transfer Queues.

Data Scheduler From the VO's point of view the Data Scheduler is a single central service. It may actually be distributed and there may be several of them, but that depends on the implementation. It accepts high-level transfer requests, where the user does not specify the source replica, or even the destination of any given transfer.

File Transfer/Placement Service The transfer and placement services may get transfer requests from the user directly or through the Data Scheduler. The difference between a transfer and placement service is only their connection to the catalog – the placement service is a transfer service that also does catalog lookup and registration of the transferred data.

Transfer Queue The transfer queue is persistent and is not tied to a site, but rather to a (set of) connections, wide area network channels (links). It keeps track of all ongoing transfers and may also be used to keep a transfer history, for logging and accounting purposes.

Transfer Agent The transfer agent again is tied to both the network channels and the VOs. The Agent may not only simply get the next transfer job out of the queue, but may also reorder the queue according to site and VO policies.

The transfer service has to be able to access and synchronize the security information between Storage Elements.

4.3. DATA CATALOGS

In the EGEE architecture, the data catalogs store information about the data and metadata that is being operated on in the Grid. The Grid Catalogs are used to manage the Grid file namespaces and the location of the files, to store and retrieve metadata and to keep data authorization information.

We decompose the catalogs into catalog feature sets which are represented by catalog interfaces (see Figure 9). These interfaces expose a well-defined set of operations to the client:

Authorization Base The basic authorization interface offers methods to set and get permissions on one or more catalog entries. The permissions are represented as a set of ACLs. The ACLs that the gLite implementation provides are read, write, execute, remove, list, set permission, set metadata and get metadata.

Metadata Base The methods of the base metadata interface deal with setting, querying, listing and removing metadata attributes to one or more catalog entries.

Metadata Schema The schema interface allows the definition attributes and group them into schemas. Attributes may be added and removed from existing schemata. These attributes are then available to be filled with values through the Metadata Base interface.

Replica Catalog The Replica Catalog exposes operations concerning the replication aspect of the grid files, connecting the logical and the storage namespaces. Operations are for example to list, add and remove replicas to a file identified by its GUID or logical file name.

File Catalog The File Catalog allows for operations on the logical file namespace that it manages. The File Catalog operations are for example making directories, renaming logical file entries and managing symbolic links.

File Authorization The methods needed to implement a standalone authorization service, as an extension to the *Authorization Base*, are in this interface, which are basically the ability to add and remove permission entries.

Metadata Catalog In order to implement a standalone metadata catalog, as an extension to the *Metadata Base* and *Metadata Schema*, this interface provides methods that are needed to add and remove entries.

Combined Catalog It usually makes sense to provide catalogs that implement the File and Replica interfaces. The combined catalog interface defines a set of convenience methods which in principle could be implemented on the client side by calling the File and Replica Catalog interfaces in sequence. If this interface is implemented by a catalog service, it either also implements File, Replica and Base Authorization interfaces or calls upon an external catalog in order to do so. If the implementation is not in the same place, the Combined Catalog implementation needs to maintain a persistent state of all operations it performs across catalogs in order to make sure that the operations only occur in a synchronized manner and that the method semantics are preserved.

StorageIndex The Storage Index and Data Location Interface interfaces are tightly coupled to the File and Replica catalog functionality. They offer methods to return the list of Storage Elements where a given file has a stored replica.

The reason for this interface decomposition is to have service interfaces with well-defined semantics which may be implemented by many parties. In this mode of operation, a possible scenario is that more than one interface is implemented by the same service. In Figure 9 the interfaces are grouped such as to show which gLite service provides what implementation. These are

gLite Fireman The gLite FIle and REplica MANager called the Fireman catalog implements all file management interfaces.

gLite StorageIndex The Storage Index service is provided also by the Fireman implementation, but is offered as a separate porttype.

gLite Metadata The standalone gLite Metadata Catalog implementation offers a full metadata catalog solution.

gLite FAS The File Authorization Service can be deployed as a simple authorization enforcement service for file access.

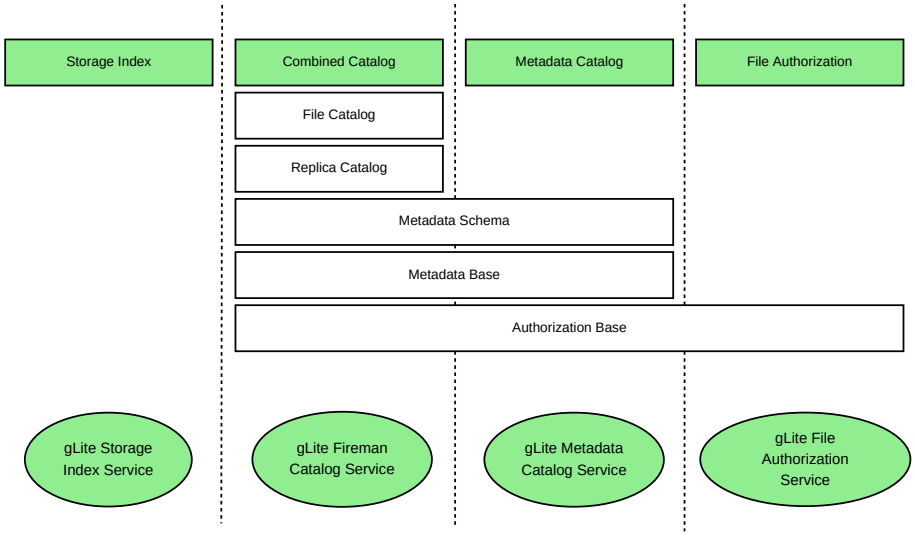

Figure 9: Catalog Interfaces with the gLite services implementing them.

Of course another implementation may choose to implement the interfaces in a different manner, to even further extend the possibilities for deployment of the services.

From the security point of view, the FAS service is tightly connected to the SE, as it is the PDP (policy decision point) for the SE. The enforcement is done by forcing the user to go through the `gLite` data access door, the gLite I/O service (this is the PEP, the policy enforcement point). There are issues with this setup as we will discuss below. The alternative is to have the SE itself as the PEP, which is an option also discussed below when we are mapping the `gLite` implementation to the security models discussed in the previous section.

4.4. Data Service Security

As mentioned in the introduction, it is important to distinguish services owned and managed by the VO from those owned and managed by the site administrators. Throughout this section, all figures are shape and color-code the components that are managed by the VO (in green circles) and by the site (in blue rectangles). Some components of the Data Movement services are managed by both, these are colored in a greenish blue and their shape is rectangular with rounded edges (e.g. in Figure 8).

The importance of this taxonomy is two-fold:

- It allows the assignment of clear responsibilities for every Grid service, which is important for accounting, auditing and liability, which are all important aspects of the security management and design.
- It allows us to identify service locality, since all site-managed services are running at a well-defined site. VO-managed services are usually 'free-roaming' services, i.e. the VO is free to choose on which site to deploy them, and vice versa each site, knowing what service is VO managed and what service is site-managed, is free to choose which VO service to deploy.

As discussed above, on the Grid it is very difficult to provide uniform security semantics for data since every Storage Element may be built on different local services with conflicting semantics. It is possible, however, to provide uniform file access semantics using the PKI infrastructure. Figure 10 shows the steps that are needed to authorize Grid file access (either through I/O or through the data movement services).

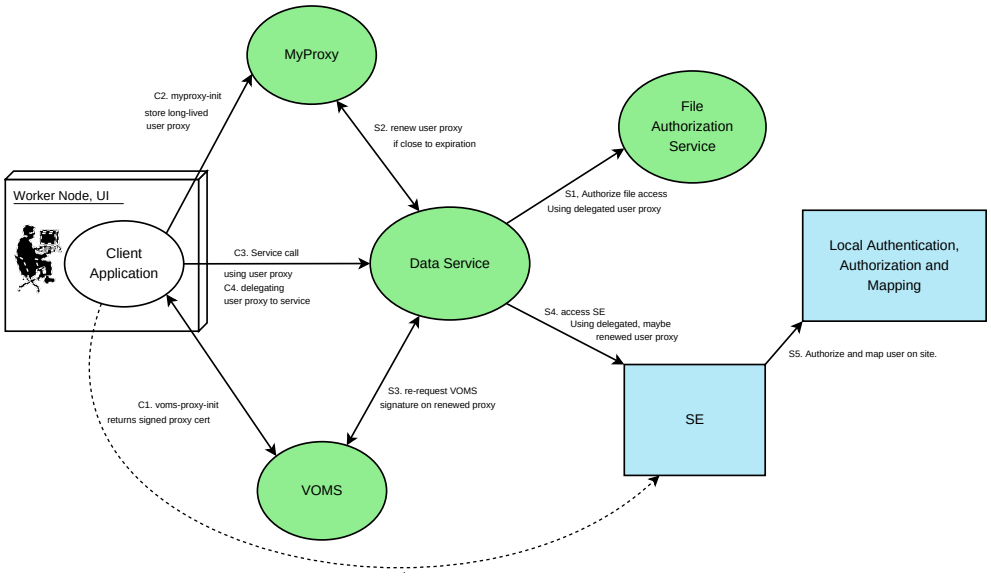

Figure 10: Security interactions of the Grid Data Services: Grid I/O and data transfer. The steps are explained in the text. The dotted line represents direct, non-Grid access.

In `gLite`, the security models that is aimed at first, is model 3 (cf. Section 3.3.) and model 5 (Section 3.4.). In this section we delve into the details of the implementation and show in detail all the interactions between the services. It illustrates the nontrivial nature of the problem.

Client

C1 The Client first needs to acquire a VOMS proxy by issuing `voms-proxy-init`. The VOMS server returns a signed VOMS proxy, with some additional fields in the proxy. The ones relevant to the data services are:

DN The DN provided in the user certificate identifies the user. Its uniqueness within its VOMS domain is guaranteed by the VOMS server.

VO The VO name is also provided as part of the VOMS proxy. This is an extension.

Groups The user may be member of one or several VOMS groups. These groups are also provided as part of the VOMS proxy extension.

C2 The user can also put a long-lived certificate into a MyProxy server. 'Normal' certificates last for a few hours (12 is the default), which may be insufficient for some tasks (like very long lasting data transfers). Therefore a longer proxy can be acquired by the user upon request. However, a long lasting proxy should not be used directly (otherwise the proxy concept would be useless) so it is kept in a MyProxy server, which uses it to extend the lifetime of short-lived proxy certificates. The lifetime of the short proxies may of course be only extended up to the total lifetime of the long-lived proxy.

C3 After having acquired a proxy and optionally having placed a long-lived one into MyProxy, the user can contact the service (either the Grid I/O or File Transfer/Placement Service), which will successfully authenticate and authorize the user if the credential is valid.

C4 Although the user may not be aware of this process, the user also delegates his credential to the server so that it can further process the request on the user's behalf.

Server

S1 After the server receives the client's request (C3), it not only checks the DN and the VO to be valid for the given server, but it also contacts the File Authorization Service (FAS) using the user's delegated credentials (from C4). It is as if the user would contact the FAS himself. The FAS will authorize the operation, i.e. whether the given file may be accessed, read or written by the user. The FAS takes the DN and the Groups of the user into account.

S2 Upon successful authorization with the FAS, the user's request may need further processing. In the File Transfer Service, it is common that requests sit on the Transfer Queue for hours. If the user's delegated proxy nears expiration, a Transfer Agent Actor will contact MyProxy and ask for a renewed proxy.

S3 VOMS signatures and attributes may need to be renewed as well, since they expire also periodically; so the same Actor also has to call VOMS to fill in the necessary attributes in order to get a fully renewed VOMS proxy. These two steps are not necessary in the synchronous Grid I/O case.

S4 Finally, if the user's certificate is valid and not expired, the server will contact the SE (either its native I/O or GridFTP or SRM interface) using the user's delegated proxy and will try to access the data.

S5 On the SE server side, it is foreseen that for each access the SE also authorizes and maps the user locally (based on the certificate it's being handed with the request) by contacting the site-local (*blue*) authorization and mapping service.

Model 3 or Model 5 The implementation as it is deployed today corresponds to model 5 because the FAS server is a central VO-controlled catalog. However, this catalog can be distributed and synchronized across sites, which is why the very same implementation may also correspond to model 3. The difficulties in synchronizing the distributed FAS **write** operations can be overcome by allowing write access only at a single FAS instance and using the locally deployed FAS instances as proxy services. This means that for checking the permissions the local service is perfectly usable, but we only have a single place for writes so that a single-master replication of the catalog is easy to achieve.

4.4.1. DISCUSSION

The data stored on some site local storage is ultimately owned by a local site account. In models 3 and 5, the site account is not mapped into a real user account but into some administrative account for the Grid services. The 'real' owner has to be identifiable by the site for accounting and auditing purposes, but it is not the storage system itself that keeps the mapping information on what real person actually owns data in the storage - this information is in the FAS, as it is the PDP. If such information would have to be logged also at the resource layer, the only possibility would be to mandate the Grid service to forward this information to the SE when contacting it. Technically it is feasible to embed the user's certificate information as attributes in the service certificate that is used to contact the SE.

As shown in Figure 7 a user can interact with all the SE interfaces using a non-grid certificate or some local account directly (*direct local access*), as if there was no Grid context *if the user is known to the system locally*. This access is represented by the dotted line in Figure 10.

However, in models 3 and 5, the PEP is in the Grid/Application layer and not in the local resource layer so the dotted line shows only a theoretical possibility. It is straightforward to prove that the direct local access is not possible in these models (hence the need for models 1 and 2). Figure 11 shows the difference between the local and Grid PEP methods.

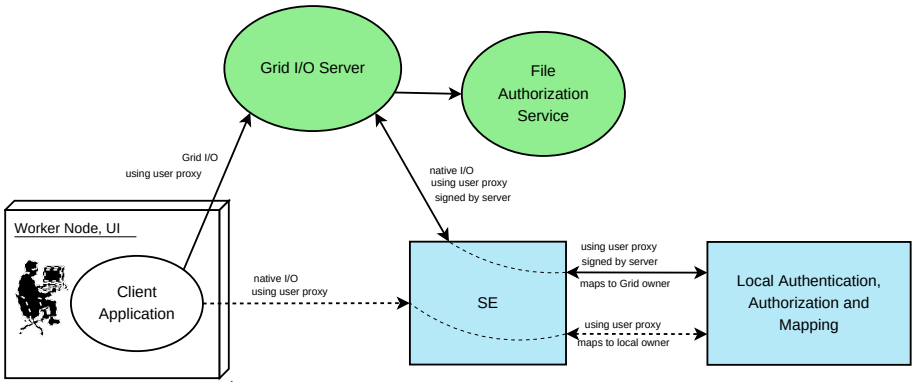

Figure 11: Certificates being used to give a user access to data either through the Grid I/O (models 3, 5) or through native I/O directly (models 1, 2, 4). In model 3 and 5, the Grid services enforce the security policy while in model 1 it's the SE itself.

So in models 3 and 5 the mode of operation is to *deny* any non-Grid access to the data once it has moved to the Grid domain, i.e. local access is not possible anymore: No local user can access their data directly through the existing native mechanisms.

If the local SE provides rich security capabilities (ACLs), the local site administrator might change the ACLs of the Grid-owned files to add individual users to the ACLs so that they can access the Grid data directly. But since this is a local administrative operation, it cannot be automatically performed by the Grid services.

4.4.2. Possible Implementations of Models 1, 2 and 4

In order to support models 2 and 4, there are two possibilities. The SE may either call out to the FAS or the user may present a token coming from the FAS proving that the operation requested can be authorized. The first possibility is straightforward - there is no need for a Grid service, however all SE operations have to call out to the FAS. The second possibility is shown in Figure 12, which may be even used in together with models 3 and 5.

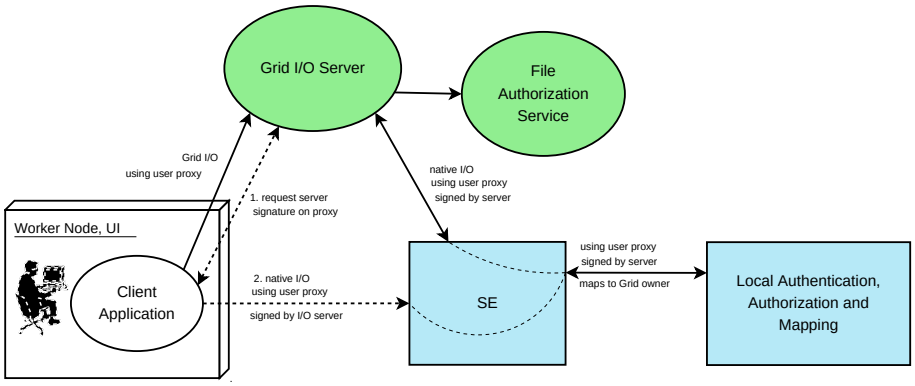

Figure 12: Certificates being used to give a user access to data both through the Grid I/O and through native I/O, mapping it always to the Grid user and authorizing through the FAS. For native I/O, the Grid I/O server needs to be contacted to provide a server-signed proxy.

In detail, this works as follows: The user identifies himself by presenting a proxy certificate (step C3 in Figure 10) and delegating it to the service (step C4). The FAS needs to *sign* the delegated user proxy

so that now the user can contact the SE with a certificate that carries the PDP information. As shown in Figure 11, the Grid I/O server contacts the SE with such a *dual* certificate. The SE again uses this certificate to contact the local mapper so that the proper local account can be selected. This mechanism allows to

- Authenticate the user locally at the site. Since the user identity is transferred with the proxy, the site has full control over who has access to its resources.
- Map the user into the proper local account. Since the service certificate or signature is part of the certificate, the mapping can be done into the proper Grid owner. If the service certificate is used with an embedded user cert, it is not a problem trusting the service and doing the mapping.

Finally, in order to implement model 1, the FAS functionality needs to be embedded inside the SE. If such SEs need to accept and adhere to the same semantics as other SEs (maybe using other models), then the 'embedded FAS' would need to synchronize its contents with the other FAS services belonging to the other Grid SEs at other sites. This is a very difficult problem, which would mandate distributed updates and multi-master operations.

5. SUMMARY AND OUTLOOK

Security in Grid data management confronts us with a series of challenges. We have presented several models that can be used to enforce uniform security semantics across the Grid for data access and storage, adhering to detailed requirements. Unfortunately none of the models discussed provides us with the silver bullet solving all issues at hand. Each of them has practical and conceptual disadvantages that are more or less important depending on the application making use of the services. What is acceptable to some is completely intolerable to others and vice versa. To find the right balance and to standardize on the necessary interfaces are the challenges that we still need to face.

The `gLite` implementation in EGEE provides already today a mature but arguably complex solution to the Grid data management security problem. We have illustrated how at least two models are implemented using the `gLite` middleware stack, and gave the background how the other models can be implemented as well. It is possible to have more than one model deployed if additional services are present that synchronize the distributed PDP information.

It will be important to investigate different synchronization techniques for PDP information and to refine and evolve the modeling of security in a complex distributed infrastructure.

6. ACKNOWLEDGMENTS

We are very much indebted to all the `gLite` Design Team for all the invaluable discussions (Predrag Buncic, Steve Fisher, David Groep, Frederic Hemmer, Kate Keahey, Maarten Litmaath, Miron Livny, Olle Mulmo, Francesco Prelz). The `gLite` Data Management development cluster has also participated actively in many of the discussions and developments, we'd like to thank explicitly Paolo Badino, Jean-Philippe Baud, Gavin McCance, Gábor Gombás and Zoltán Farkas. We also would like to thank the EGEE Biomedical Data Management group (Johan Montagnat, Christophe Pera, Daniel Jouvenot) who made us understand the real-world requirements for secure applications.

REFERENCES

[1] The Grid: Blueprint for a New Computing Infrastructure (2nd Edition). *Foster, I. and C. Kesselman, eds.* Morgan Kaufmann 2004.

[2] The Grid 2, Chapter 21. Security for Virtual Organizations: Federating Trust and Policy Domains. *Frank Siebenlist, Nataraj Nagaratnam, Von Welch and Clifford Neuman*. Morgan Kaufmann 2004.

[3] A National-Scale Authentication Infrastructure. *Butler, R., et al.,* IEEE Computer, 2000. 33(12): p. 60-66.

[4] Authentication and Authorization Mechanisms for Multi-domain Grid Environments. *L. Cornwall, J. Jensen, D.P. Kelsey, A. Frohner, D. Kouril, F. Bonnassieux, S. Nicoud, K. Lorentey, J. Hahkala, M. Silander, R. Cecchini, V. Ciaschini, L. d'Agnello, F. Spataro, D. O'Callaghan, O. Mulmo, G.L. Volpato, D. Groep, M. Steenbakkers and A. McNab*, Journal of Grid Computing (2004) p. 301-311.

[5] EGEE Middleware Architecture, *EGEE Middleware Activity JRA1*. `https://edms.cern.ch/document/594698/`

[6] EGEE Global Security Architecture, *EGEE Security Activity JRA3*. `https://edms.cern.ch/document/487004/`

[7] Site Access Control Architecture. *EGEE Security Activity JRA3*. `https://edms.cern.ch/document/523948/`

[8] The PERMIS X.509 Role Based Privilege Management Infrastructure. *Chadwick, D.W. and A. Otenko.* in 7th ACM Symposium on Access Control Models and Technologies. 2002.

[9] The GGF Grid Storage Resource Manager Working Group `http://sdm.lbl.gov/gsm/documents.html`

[10] The Open Grid Services Architecture, Version 1.0, *GGF OGSA WG*, `https://forge.gridforum.org/projects/ogsa-wg`

[11] The TLS Protocol Version 1.0, *Dierks, T. and C. Allen,* `http://www.ietf.org/rfc/rfc2246.txt`

[12] R. Alfieri , R. Cecchini , V. Ciaschini , L. dell Agnello , Á. Frohner , A. Gianoli , K. Lőrentey , and F. Spataro VOMS, an Authorization System for Virtual Organizations. In *Grid Computing, First European Across Grids Conference*, February 13-14, 2003

[13] An Internet Attribute Certificate Profile for Authorization, *Farrell, S. and R. Housley*

[14] Globus: A Metacomputing Infrastructure Toolkit. *Foster, I. and C. Kesselman* International Journal of Supercomputer Applications, 1998. 11(2): p. 115-129.

[15] A Security Architecture for Computational Grids. *Foster, I., et al.* in 5th ACM Conference on Computer and Communications Security. 1998.

[16] Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective `http://www.globus.org/toolkit/docs/4.0/security/GT4-GSI-Overview.pdf`

[17] GridFTP Protocol Specification, *W. Allcock et al*, Global Grid Forum Recommendation GFD.20, March 2003

[18] Globus Project, Grid Security Infrastructure (GSI),
http://www.globus.org/security/

[19] Globus Project, Globus Project Technical Papers,
http://www.globus.org/research/papers.html

[20] IBM, Web Services Secure Conversation Language (WSSecureConversation) Version 1.0, December 18.

[21] IBM, Web Services Trust Language (WS-Trust)

[22] IBM, Microsoft, and VeriSign, Web Services Security Language (WS-Security),

[23] Kerberos: An authentication service for computer networks, *Neuman, B.C., Ts'o, T.*, IEEE Communications Magazine 32(9), 33-88, 1994

[24] Kerberized Credential Translation: A Solution to Web Access Control, *Olga Kornievskaia, Peter Honeyman, Bill Doster, and Kevin Coffman*, February 2001. [USENIX Security Symposium, Washington, D.C. (August 2001)]

[25] Public Key Cryptography for Initial Authentication in Kerberos
http://tools.ietf.org/wg/krb-wg/draft-ietf-cat-kerberos-pk-init/

[26] A Community Authorization Service for Group Collaboration. *Pearlman, L., et al.* in IEEE 3rd International Workshop on Policies for Distributed Systems and Networks. 2002.

[27] X.509 Proxy Certificates for Dynamic Delegation *Welch, V., et al.*

[28] Use of SAML for OGSA Authorization, *Welch, V., et al.,*
http://www.globus.org/ogsa/security

[29] 19. WS-I, Web Services Interoperability (WS-I) Interoperability Profile 1.0a.

[30] *An Internet Attribute Certificate Profile for Authorization, RFC3281*
S. Farrell, R. Housley; April 2002,