Semantic Access Control for Medical Applications in Grid Environments*

Ludwig Seitz, Jean-Marc Pierson, and Lionel Brunie

LIRIS, INSA de Lyon
7, av. Jean Capelle, 69621 Villeurbanne cedex, FRANCE
{ludwig.seitz, jean-marc.pierson, lionel.brunie}@liris.insa-lyon.fr

Abstract. Access control is the field of security which deals with permissions to access resources, where resources may be computing power, storage capacity and data. On the other hand computational grids are systems, where users share those resources in a mostly transparent way. Grid access control poses novel challenges, since the distributed nature of grids make it difficult to manage access control by a central authority. Numerous overlapping domains with different access control policies exist and the sharing of storage resources makes it possible that data leaves the domain of its owner. To enable the owner to enforce his access control policy in such cases, access control solutions adapted to grid environments are needed. In this article we introduce Semantic Access Certificates as an extension to existing access control solutions for grids, to solve some problems that arise when grids are used to process medical data.

1 Introduction

Grid computing [1] is becoming a very popular solution for researchers looking for vast storage and computing capacity. A *computational grid* is a set of computing elements and data storage elements, heterogeneous in hard- and software at geographically distant sites which are connected by a network and use mechanisms to share *resources* as computing power, storage capacity and data. The usage of those computational grids is called *grid computing*.

Different interested parties like high-energy physics, terrestrial observation and genome decoding have recognized the potential value of such tools (see [2], [3] or [4] for application examples).

Another field of application is to use grid infrastructures for medical data processing (see [5], [6]). Goals of such infrastructures can be to provide researchers with a broad spectrum of data for analysis and to make it possible for patients to gain access to their data regardless where they are and where it is stored.

It is clear that in medical data processing privacy protection needs to be enforced much more rigorously than in other application areas. Moreover, the

* This work is partly supported by the Région Rhône-Alpes and the French ministry for research ACI-GRID project (http://www-sop.inria.fr/aci/grid/public/)

users of such systems (patient, medical staff) will mostly not be trained in computer security and thus need easy to use and mostly transparent services. Thus we need tools for classical security tasks like *authentication*, *data integrity*, *private communications* and *access control* in grid environments. Authentication is the task of proving ones identity over a computer network. Data integrity means to ensure that data has not been fraudulently manipulated, which can be done through *electronic signatures*. In private communications, involved parties want to communicate confidential data over an insecure channel, here *encryption* is the technique that is most commonly used. These fields of security are well covered in common applications and have been adapted to grid environments without major changes (see [7] or [8] for examples).

This article focuses on access control for medical data in grid environments. We define access control as the domain of security, which deals with the permissions of *entities* to access *resources*, where entities may be users, groups of users, programs, daemons etc. and resources may be computing power, storage capacity and data. Unlike the other security aspects the structure of the grid poses novel problems for access control that can not be dealt with by classical solutions. These problems arise from the distributed nature of grid users and grid resources, where user activity may cross organizational boundaries, and data may be stored outside the user's domain of control. This also means that users can have to handle the access rights of data belonging to the same semantic context but physically distributed on different storage elements. As we consider the patient to be the owner of his data in our medical application, flexible mechanisms are needed, that permit the patient to manage the rights of his data, even when he is not connected to its storage site.

The structure of the article is as follows: In section 2 we describe the access control problems, arising specially in medical applications on grid environments. We then extract some desirable functionalities of an access control system for those applications. In section 3 we present classical approaches to access control while section 4 gives grid specific solutions. In section 5 we propose an extension to the existing grid mechanisms, designed to provide the desirable functionalities defined in section 2: access control through *Semantic Access Certificates*. Finally we summarize and give directions of our future work in section 6.

2 Medical Data on Grids

We present a somewhat constructed but still realistic use-case, to illustrate a class of access control problems we want to address in this article.

Imagine a patient who visits a new doctor (for instance a patient in vacation visiting a doctor far from home). He wants to give this doctor access to some of his medical data, for example radiographs made at a certain hospital and the medical files of his former doctor. He doesn't know the storage element, neither does he know the filenames or directories where his data is. The patient only wants to interact with the doctor and not with the storage site to grant the access. If a large number of medical files are concerned, he would prefer to grant access to all those files at once through some kind of classification mechanism, and not separately for every single file.

On the other hand the doctor wants to use the rights given to him to locate and retrieve the data. If the treatment is done by a team of doctors, all team members should be able to use the access rights. The doctor also needs to be able to access future data of the patient, necessary for treatment and diagnosis. He doesn't want to have to take difficult security measures to protect the permissions that were given to him by his patients, since this is not his specialty.

From this example we extract the following list of desirable functionalities specific to medical data processing.

- *Group granting:* It should be possible to specify permissions for groups of doctors, thus a patient could give medical teams access to necessary data in one operation.
- *Class granting:* It should be possible to specify permissions to classes of data, to enable the patient to give access to all data concerning a certain field of medicine in one operation.
- *Storage location independence:* Patients should be able to specify permissions without exact knowledge of the resource storage sites, because a system that requires the patient to keep track of his medical data and remember how to index it in the system would neither be accepted nor would it be usable.
- *Offline delegation:* Patients should be able to assign permissions offline, without being connected to the storage sites. Since the patient is the owner of his medical data, and the storage site is mostly some kind of laboratory or clinic it would be cumbersome, if the patient had to contact the storage site, every time he wants to grant access to some of his data.
- *Personalized permissions:* The assigned permissions should be bound to a specific entity or group of entities, for a given amount of time and not be usable by unauthorized entities. This minimizes the impact of loss or theft of those permissions.

In the two following sections, we explain why classical access control schema do not address completely all these problems before giving a possible solution through enriched access certificates.

3 Classical Access control

The UNIX style access control [9] with read, write and execute permissions and its user, group and other classification has the benefit of simplicity. However it does not provide any functionality to deal with distributed data of the same class. Access rights management has to be done online and there is no way to define specific access rights for a user that has no account on the system.

Using *access control lists (ACL)* [10] adds some flexibility, however it is a local access control system as well and does neither provide offline access rights granting and distributed data classes management.

Role based access control (RBAC) [11] groups entities and resources by so called *roles*. Those are collections of entities and access rights grouped together based on different tasks they perform in the system environment. Entities are given the right to take certain roles and thus use their permissions, to be able to use the resources that are necessary for their tasks. The RBAC approach permits to group access rights by semantic meanings, but to deal with distributed data, extensions of the classical RBAC approaches are needed. Also pure RBAC is somewhat non flexible for the granting of specific rights, since it only permits to grant rights by defining an appropriate role and assigning users the right to use it.

One such extension, that aims specifically at distributed systems is *OASIS* [12],[13]. OASIS differs from other RBAC schemes in three major ways. First the role management is decentralized, which means that every service in the distributed system may define their own roles and control the conditions how those can be used. The services interoperate through Service Level Agreements (SLA). Second OASIS permits to parameterize roles, which allows to define generic roles (e.g. A is doctor of B) were the parameters can be chosen for specific applications. Third it replaces priviledge delegation by a mechanism called *appointment*. Certain roles have the function to permit the issuing of appointment certificates, that permit to activate roles together with an authentication mechanism. The OASIS approach permits to manage access control for distributed data of the same semantic class, however it does not permit offline granting of access rights, since one has to activate an appointment role to grant rights.

4 Grid access control solutions

This section deals with today grid solutions for access control. We present three common grid computing environments, Condor [14], Legion [15] and Globus [16] and we argue why their solutions have to be extended with further functionality for medical grid applications.

Condor [14] is a specialized workload management system for compute-intensive jobs. Its resource management mechanisms are similar to UNIX style access control. Main differences are some additional modes of access besides traditional read and write permissions. Also the classical meanings of the read and write permissions are slightly changed, since all permissions are not relative to files, but to the Condor system itself. Therefore read access means the permission to access information from Condor, write to change settings etc.

The Legion project [15] uses an object oriented approach to grid computing. Thus resources, as files, services and devices are considered as objects and access to those are through functions of these objects. The Legion access control approach is that each object is responsible for enforcing own access control policy. Thus each object has a *MayI* function, invoked before any other functions of the object may be called. This function allows resource owners to define their own access control mechanisms. A default MayI implementation exists that is based on ACL's and credential checking (authentication).

The Globus grid toolkit [16] proposes mechanisms for translating users grid identities into local identities. This would allow users to sign onto the grid and then use resources without further authentication (single sign-on). The benefit for local resource providers would be that through the translations mechanism, local access control policies can be enforced.

More recently Pearlman et al.[17] have also been working on an access control system that relies on a Community Authorization Service (CAS). The idea is that users shall be able to request access right at a CAS server, which will assign them capabilities, based on the request and the user's role within the community. The user can present those capabilities at a resource server to gain access on behalf of the community. Thus this approach integrates the benefits of RBAC into a grid access control system.

All approaches described above lack some mechanisms for managing access to distributed data belonging to a common semantical class and they do not provide mechanisms for offline granting of rights. Therefore they have to be extended for use in our medical grid applications.

5 Semantic access certificates

A number of classical approaches exist that grant access to data through the use of electronic documents, like the Kerberos V5 protocol [18] access tickets, however these approaches do not permit the management of transparent access to distributed data, since the storage site has to be known, to issue a valid ticket. We have therefore decided to enrich the concept of access tickets to get rid of those limitations.

5.1 The structure of a semantic access certificate

We will use a doctor/patient scenario as in our motivation to describe the structure and use of access certificates, this does of course not mean, that we limit ourselves to use in this context, but serves to clarify the relations between the involved entities.

We first emphasize the medical data we want to give (or refuse !) access to. Upon creation of a medical data file, it will be assigned a certain set of meta-data. This set necessarily contains an unique identifier of the patient that owns the data (since one of our assumption is that one data belong to one patient), the date of creation, and a hash of the data signed by the patient to guaranty integrity.

Additionally, data can be associated to classes, to allow for more coarse grained access control decisions. In medical applications, classifying data by medical specialty allows a patient to give access rights to a doctor for the specific class of data that is needed for his treatment or diagnosis. A patient visiting a dentist should give him access to all dental data and medicine allergy, while not giving access to other classes of information such as psychological analysis or lung radiographs. Some ontology of the medical domain should allow for a fine classification of the data produced at different medical services (but note here that such an ontology does not currently exist globally, but this research field is very active today, see the US Unified Medical Language System for instance [19]).

We define *Semantic Access Certificates (SAC)* as electronic documents that contain the following information (see figure 1):

- an unique identifier of the doctors that are granted access,
- an unique identifier of the patient granting access,
- an unique identifier of the data or class of data to which access is granted,
- the allowed modes of access,
- a validity period,
- and the electronic signature of the entity granting the rights.

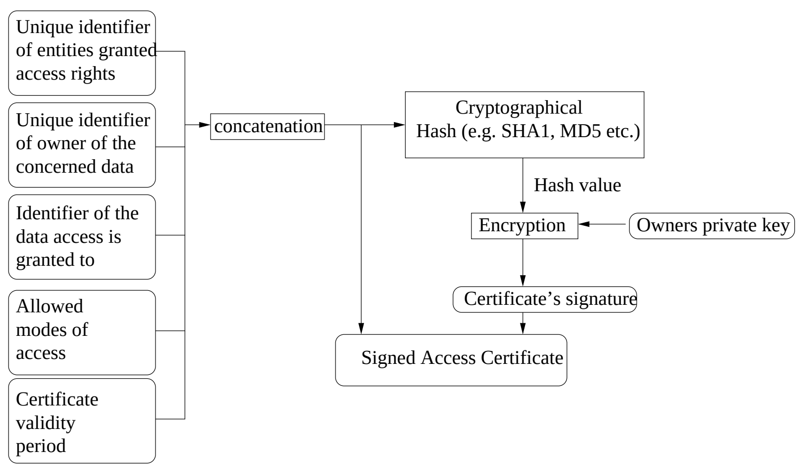

Fig. 1. Components of a signed access certificate

Let us now detail and argue each of the fields present in this certificate.

First appears the doctor (or the group) for whom the certificate has been issued : For a single doctor its public key may be used as unique identifier, management of groups of medical staff is done through trusted authorities that issue group membership certificates.

Similarly, the unique identifier of the patient may be its public key.

Creating unique identifiers for files can be done by concatenating the patient identifier with a hash (MD5, SHA1, ...) of the file content (or the semantic class name). The semantic value transported with the certificate when classes of access are given that way allows for a much more flexible and easy to use system. The users can thus effectively give access to entire part of his medical data. The classification of data allows to give access rights based on semantical criterions, as it is independent of storage sites, even access to distributed data of the same semantical context may be managed this way.

Fourth, the allowed access mode should be as simple as "read", "write" or "read/write" or more complicated such as "read anonymised" (for instance when an user wants to give access to some of its medical information to medical researchers for epidemiologic studies) or "append information" (when an user wants the doctor to add new information to its medical file). The handling of these different access mode lies beyond the scope of this article.

The validity period in access certificates is generally set so that a hacker stealing the certificate won't be able to decode it and use it before it expires. We see another interest for us : a patient should want a doctor to access his medical data during a given period of time (for instance while in the doctor office or for the treatment duration, but not longer).

Finally, the electronic signature allows classically for the integrity of the issued certificate.

To revoke SACs the patient will be able to issue a SAC revocation request, that uses the identifiers of the patient and the doctor holding the SAC with the signature to form an unique identifier of the SAC. This revocation request will be broadcasted to the resource server sites, in a similar way as certificate revocation lists.

As the patient owns his data, he is the only one that can issue access certificates or change the data. A copy of the unique file identifiers of all his data need to be stored with him, so he can issue access certificates to specific data files. If the data is associated to a class, the classification also needs to be stored with him. For the revocation mechanism, the unique identifiers of the certificates need to be stored with the patients too. Such information will be stored on a smartcard or a similar cryptographic token along with the patients public/private key pair. Those tokens will be kept by the patients like today medical insurance cards.

5.2 Usage of SAC

Let's imagine a scenario where a doctor has successfully acquired a valid SAC and wants to access some data. The process would have the following actors: The doctor requesting some data, the doctors grid interface, a resource broker mechanism provided by the grid infrastructure to locate resources and resource server sites that offer access to resources.

The request formulated by the doctor would be translated by his grid interface into a grid readable format, like for example a request to the Globus GASS system [20]. The exact format of the request lies beyond the scope of this article and shall thus not be treated any further.

The whole process that follows the submission of a request should remain transparent to the doctor, until the request results in either success or definitive failure.

Figure 2 illustrates the use of access certificates and the protocol is specified here:

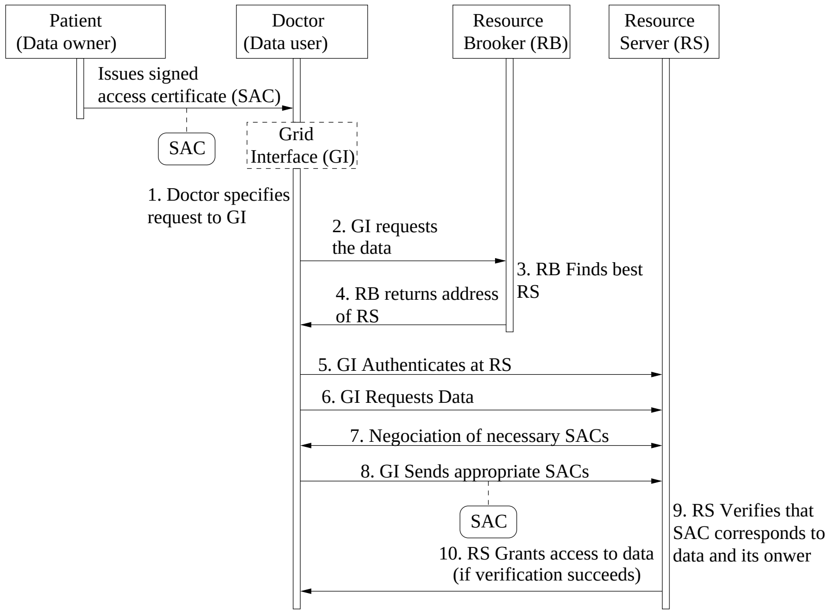

Fig. 2. Usage of access certificates

1. The doctor specifies what data it wants to his grid user interface.
2. The doctor's grid interface contacts the resource broker and submits the data request.
3. The resource broker locates the best resource server[1] that has the requested data available.
4. The resource broker transmits the address to the doctor's grid interface.
5. The doctor's grid interface authenticates at the resource server.
6. The doctor's grid interface transmits the data request.
7. The doctor's grid interface and the resource server negotiate which access certificates are needed for the requested access. The negotiation is carried out in the following steps:
   (a) The doctor's grid interface submits the request to the resource server
   (b) The resource server locates the requested data's meta-data section and extracts the patient's unique identifier and the data classification if one exists. It then returns those to the grid interface.
   (c) The grid interface uses the provided meta-data to scan the doctor's access certificate repository for appropriate access certificates and returns those to the resource broker.

[1] The determination of the best server could be done by various criterions like: the one with the fastest connection, one in the same country for legal reasons, a trusted site etc.

8. The doctor's grid interface sends the appropriate access certificates to the resource server.
9. The resource server checks that the access certificates bears a valid signature by the patient owning the data.
10. If access can be granted the resource server gives access to the grid interface, otherwise it returns an appropriate request failure message.
11. If the access is granted the grid interface proposes the access to the doctor, if the request failed due to insufficient permission the doctor is notified, if the request failed for other reasons, the grid interface resubmits the request to the resource broker, excluding the addresses of the previously contacted resource servers in this new request.

In section 2, we have exhibited five desirable functionalities of access control in medical area. We now address their adequation with the SAC definition and protocol. The SACs allow directly, from their definition, for *Group granting* and *Class granting*, a certificate being issued for a doctor or group of doctors and for a data or a given class of data). *Storage location independence* is assured since the identifier depends only on the patient and the content of the data. Thus any replica of the data stored on the grid can be accessed with the same SAC. Access certificates can be granted offline (*Offline delegation*) i.e. without being connected to the resource server site. The patient simply has to prepare the electronic document in the right format, to assure he has the right identifiers and to sign it digitally. From this point on, the certificate is bound to the doctor or group of doctors (*Personalized permissions*) it was issued for and can be transmitted even through an insecure channel. The electronic signature of the patient ensures its integrity and thus no unauthorized person can use the SAC.

6 Conclusion

We have proposed in this article a novel approach to access right management on grids, in the medical data field. Our contribution is the definition of a Semantic Access Certificate (SAC), holding the access control policy not only based on roles (patient, doctor, nurse, etc.) but also on the semantic of the stored data (the specialty it belongs to). While being illustrated here in the medical field, the proposed schema can be used in any application where privacy protection have to be enforced.

However this article only covers a part of the access control problems arising in medical data processing on the grid. Special modes of access like anonymized read for medical research, problems of access logging and data persistence are not treated in this article. Therefore it will need to be expanded and should not be used alone. We are currently working on the design of reactive documents that will carry their special access mode programs, access logging and perenity enforcement mechanisms with them. These will allow the users to specify their own access modes, logging and perenity conditions and to enforce them, even if the data is stored at untrusted sites.

References

1. Foster, I., Kesselman, C., eds.: The Grid Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, Inc. (1999)
2. European Data Grid: The datagrid project. http://eu-datagrid.web.cern.ch/eu-datagrid/ (2001)
3. Cactus Community: The cactus code. http://www.cactuscode.org/ (2002)
4. National Institute of Advanced Industrial Science and Technology: Grid technology research center. http://www.aist.go.jp/ (2002)
5. MEDIGRID: Medical data storage and processing on the GRID. http://creatis-www.insa-lyon.fr/MEDIGRID (2002)
6. University of Manchester: Manchester Visualization Centre: ix-grid. http://www.sve.man.ac.uk/mvc/Research/iX-Grid/ (2002)
7. Foster, I., Kesselman, C., Tsudik, G., Tuecke, S.: A security architecture for computational grids. In: Proceedings of the 5th ACM Conference on Computer and Communications Security Conference. (1998) 83–92
8. Ferrari, A., Knabe, F., Humphrey, M., Chapin, S., A.Grimshaw: A flexible security system for metacomputing environments. In: Proceedings of the High Performance Computing and Networking conference 99 (HPCN'99). (1999)
9. Nemeth, E., Snyder, G., Seebass, S., Hein, T.: UNIX System Administration Handbook (3rd Edition). Prentice Hall PTR (2000)
10. Lampson, B.: Protection. In: Proc. of the 5th Princeton Conf. on Information Sciences and Systems, Princeton, 1971. Reprinted in ACM Operating Systems Rev. Volume 8, 1. (1974) 18–24
11. Sandhu, R.: Role-based access control. In: Advances in Computers. Volume 46. Academic Press (1998)
12. Yao, W., Moody, K., Bacon, J.: A Model of OASIS Role-Based Access Control and its Support for Active Security. In: Proceedings of Sith ACM Symposium on Access Control Models and Technologies, SACMAT. (2001) 171–181
13. Bacon, J., Moody, K., Yao, W.: Access Control and Trust in the use of Widely Distributed Services. In: Lecture Notes in Computer Science. Volume 2218. Springer (2001) 295–310
14. Condor Team of the University of Wisconsin: Condor, high troughput computing. http://www.cs.wisc.edu/condor/ (1988)
15. Legion Research Group of the University of Virginia: Legion, a worldwide virtual computer. http://legion.virginia.edu/ (1993)
16. Globus Project: Globus toolkit. http://www.globus.org/ (1998)
17. Pearlman, L., Welch, V., Foster, I., Kesselman, C., Tuecke, S.: A community authorization service for group collaboration. In: Proceedings of the 2002 IEEE Workshop on Policies for Distributed Systems and Networks. (2002)
18. Kohl, J., Neuman, C.: The Kerberos Network Authentication Service (V5). Technical report, The Internet Engineering Task Force IETF (1993) http://www.ietf.org/rfc/rfc1510.txt.
19. UMLS: Unified medical language system. (http://www.nlm.nih.gov/research/umls/)
20. Bester, J., Foster, I., Kesselman, C., Tedesco, J., Tuecke, S.: Gass: A data movement and access service for wide area computing systems. In: Proceedings of the Sixth Workshop on I/O in Parallel and Distributed Systems. (1999)