A Trusted Data Storage Infrastructure for Grid-based Medical Applications[*]

Guido J. van 't Noordende, Silvia D. Olabarriaga[†], Matthijs R. Koot, Cees Th.A.M. de Laat
Informatics Institute, University of Amsterdam, The Netherlands
[†]Academic Medical Center, University of Amsterdam, The Netherlands

Abstract

Most existing Grid technology has been foremost designed with performance and scalability in mind. When using Grid infrastructure for medical applications, privacy and security considerations become paramount. This leads to a re-thinking of implementation and deployment aspects of common components of the current Grid architecture. This paper describes the impact of privacy and security considerations on the Grid infrastructure design, and enumerates trust aspects which must underpin the design of Grid technology to support medical applications. We propose a novel security framework for securely handling privacy sensitive information on the Grid.

1 Introduction

Most Grid technology to date has been designed for high-performance and high-throughput computing and data storage [6, 16]. Initially Grid applications aimed mostly at the Physics community, but recently many other domains, such as (computational) Biology, Pharmaceutics, and Medical research, have shown increasing interest in Grid infrastructure. Current Grid middleware, including gLite [3] or the Globus Toolkit [4], hide many aspects such as data distribution and replication from users of the system. As a result, jobs and data are often transferred through multiple Grid components in different administrative domains in implicit ways without the awareness of the end-user. Medical applications, however, have strict requirements on secure data handling and storage due to data privacy concerns. Therefore, middleware intended for usage in the medical domain should be extended to support (application-defined) policies that define where particular data may be stored, in what form, and which jobs from which users may access this data from what hosts or administrative domains.

[*]This research is part of the Dutch program Virtual Laboratory for e-Science (VL-e), supported by a BSIK grant from the Ministry of Education, Culture and Science and the ICT innovation program of the Ministry of Economic Affairs.

We present a new framework that allows for explicit control of aspects related to data access and distribution in Grid systems. It makes a clear distinction between data storage components, access control, and job authentication aspects, and includes auditing for data related operations. The paper is organized as follows: first we describe a use-case for medical research, based on our own experience [21]. Next, we analyze legal requirements with regard to medical data and technical aspects that are relevant when using Grid infrastructure to manage privacy-sensitive data. Finally, we describe a framework that allows data owners to express fine-grained data distribution and access control policies to allow for more secure handling of medical data on the Grid.

2 Usage Scenario

Figure 1 shows a typical Grid infrastructure deployment for medical research. A Grid storage system in one trusted administrative domain is used for (de-identified) medical research data. Although data is often replicated across different domains to enhance availability and reliability, we here assume that all storage facilities reside in administrative domain trusted by the data owner. We refer to the storage infrastructure as a Storage Resource Broker (SRB) in a general way. Different incarnations of storage resource brokers exist (e.g., SDSC SRB and dCache [1]).

First, Researcher A (data owner) uploads the data into the SRB. Researcher B can now submit a job on the Grid through a Compute Resource Broker (CRB) which can reside in any administrative domain. The CRB transparently selects a cluster, typically based on load, where the job is scheduled for execution. The user controls job submission via some *job description*, e.g., using a Job Submission Description Language (JSDL [5]), which describes the binary to execute on the compute element and input files. In addition, the job description can specify a specific cluster, or resource requirements, to be matched with available Grid resources prior scheduling. Running jobs can access files that the job's owner is authorized to access. In some cases, the Grid middleware *pre-fetches* required input files using the job owner's credentials prior to job execution.

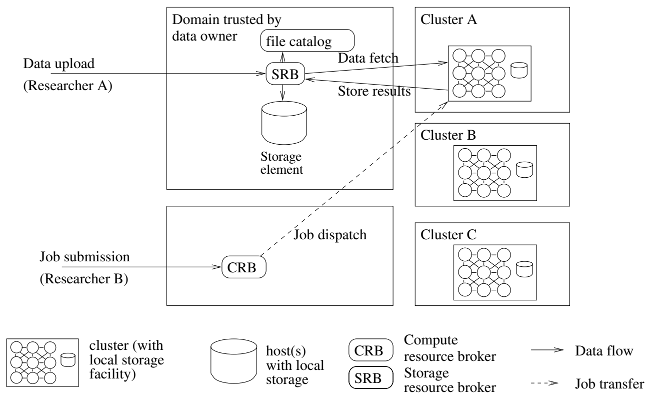

Figure 1. Use-case for medical imaging research showing different administrative domains of grid resources with emphasis on data and job flow. See text for details.

Fig. 1 also shows a File Catalog that provides a mapping between Grid 'logical file names' and the underlying physical files, which may be replicated on different storage systems on the Grid. Additionally, an SRB may also maintain a metadata service (not shown). Since metadata and file names may contain privacy sensitive information, both services should be managed by a trusted domain. Although this example considers files, the concepts explored in this paper apply equally well to, for example, database systems.

3 Legal Requirements

Here we focus on the European and Dutch laws and regulations for handling medical data; for more information about other countries see [14, 2, 17, 12].

The European Union (EU) has produced legislation on handling personal information and privacy [2]. Countries outside the EU have adopted or are adopting legal measures to allow exchange of personal data with the EU countries (e.g., [8]). The EU regulations can be seen as leading guidelines for handling personal data [14]. The data protection regulations can be summarized as follows. First, there must be a *necessity* for data collection and processing. Related to that, for each data collection, there has to be a clear *purpose binding* which specifies what is done with the information. Usage of data beyond this specified purpose is not allowed. In addition, a *minimality principle* exists, which states that only the minimum information for the required purpose may be collected. Furthermore, there has to be *transparency of personal data processing and collection*, implying that the data subject is informed of data collection (opt-in or opt-out) and that the data subject has a right to access the information. Finally, the regulations require that information is *accurate*, which implies that the information must be kept up-to-date [14].

Two Dutch laws [9, 10] formalize what may be done with data collected from a patient in the course of treatment. In general, usage of patient information outside the scope of the patient's treatment is not allowed, unless there is considerable public interest or similar necessity to do so. Medical scientific research is often considered such an exception [17]. If a patient explicitly consents with usage of his data for medical research, that data is purpose-bound to a specific medical research activity. The data may not be disclosed beyond this activity. The physician or medical researcher who determines the purpose and means of processing is legally responsible for ensuring an appropriate level of security to protect data. The restrictions described above *only* apply to personal data. In some situations, the data can be de-personalized to circumvent these restrictions, e.g., as done in [18, 20, 13]. However, complete de-identification is hard to get right, and re-identification is often possible [22, 19]. For this reason, de-identified information should be considered confidential, and appropriate distribution and access control mechanisms are required.

4 Basic assumptions throughout this paper

The Grid Security Infrastructure (GSI, [15]) provides the basic user and host authentication facilities used by most mature Grid infrastructure implementations. GSI essentially comprises a Public Key Infrastructure (PKI) that is used to sign *user identity* and *host* certificates. Users can create limited-lifetime *Proxy certificates* which allow them to send credentials with their jobs for authentication, without the risk of compromising the user's private key. Proxy certificates are used for all transactions by a job, such as gridFTP transactions. We here assume that all authorization decisions with regard to data are based on GSI user identities by means of Proxy certificates. Other approaches (such as role-based or attribute-based authorization, as proposed in [11]) are possible, but not required for our framework. Many Grid infrastructures manage access control to resources and storage based on virtual organization (VO) membership information. However, VO-based authorization is often too course-grained for protecting medical information: there may be many users (e.g., researchers) in a VO, which may not all be equally trusted to access particular data sets. Therefore, we require authorization based on user identities.

We assume that the implementation of a job is trusted when this job's owner is trusted. In particular, we assume that medical researchers are aware of confidentiality aspects regarding medical data and treat this data as confidential information (see sec. 3). In the proposed framework, jobs can only access data from hosts that are trusted by the data owner, and we assume that the job submitted by a trusted user will not leak information to unauthorized parties. However, even trusted hosts in general cannot control how information is treated by jobs while they execute [25]. Therefore, jobs do not ship (output) data back to the (possibly untrusted) CRB through which the they entered the system, but they store any sensitive output data only secure storage, preferably the system that contained the input data.

5 Problem Analysis

Grids are, by nature, distributed across multiple administrative domains, a few of which may be trusted by a specific data owner. Software and middleware typically run on operating systems (OS), such as Linux, that allow administrators to access all information on the system. These systems might also not be well protected against physical attacks, such as stealing hard disks. These aspects should be part of a *risk assessment* done when decisions are made on which sites are trusted to store or access particular information.

Given legal constraints, trust decisions will and should be conservative. For example, unencrypted data, file names, and other sensitive metadata should only be stored in trusted domains, e.g., in the hospital. This aspect is even more prevalent in systems where jobs on remote machines can access medical data. Current OSs such as Linux provide little assurance that information stored on the system cannot be leaked to external parties [25]. Even if files are removed after the job exits (e.g., temporarily created files), the contents could be readable by administrators or possibly attackers while the job executes. Furthermore, disks may contain leftover information from a job's previous execution, which is readable by an attacker who gains physical access to a storage device, if the system is not properly configured [7]. As another example, it is possible to encrypt swap space in a safe way, but this is an option that has to be explicitly enabled in the OS. For these reasons, it is important for a data owner to identify critical aspects of the administration and configuration of a remote host, before shipping data to a job running on that host.

Another problem is that a data owner cannot control nor know the trajectory that a job took before it was scheduled on a host, since this is implicit and hidden in current middleware. Therefore, even if the host from which a job accesses data is trusted by the data owner, there is a risk that the job was manipulated on some earlier host. Additionally, current middleware does not provide a way to securely bind jobs to Proxy certificates: a certificate or private key bundled with a program can easily be extracted and coupled to another program which pretends to be the original program. In Grids,

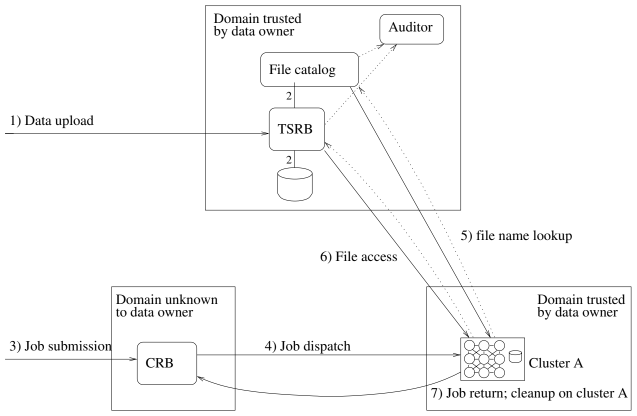
Domain trusted by data owner
Auditor
File catalog
2
1) Data upload
TSRB
2
5) file name lookup
6) File access
Domain unknown to data owner
Domain trusted by data owner
3) Job submission
CRB
4) Job dispatch
Cluster A
7) Job return; cleanup on cluster A

Figure 2. Proposed framework: files, file names and metadata are managed by a Trusted SRB. Dotted lines depict microcontract establishment and auditing, solid lines depict data flow and job transfers. See text.

this issue is exacerbated by the fact that a job may traverse several middleware processes in different domains before it is scheduled at some host. Each of these hosts or domains may be malicious, and the administrator or an attacker that gains access to one of these hosts may replace the original job's binary or initial script with another program that leaks information to an external party. Other authentication schemes (e.g., username/password based) do not improve this situation.

Summarizing, a number of implementation issues should be solved before we can ensure that possibly re-identifiable information cannot be accessed by unauthorized parties. First, a secure binding between jobs and Proxy certificates must be provided. Second, a data owner should be able to express in a policy which administrative domains he or she trusts to handle privacy sensitive information in a safe way with regard to integrity and confidentially, based on a risk assessment. Third, a data owner should be able to express policies with regard to a remote system's configuration details which are relevant to the way in which data is handled.

6 Proposed Framework

We propose a framework for secure handling of privacy sensitive information on Grids that allows for controlling data access and distribution aspects. The components and interactions of the framework are presented in Fig. 2.

The framework is centered around a secure storage infrastructure called *Trusted SRB (TSRB)*. There may be many TSRBs on the Grid, possibly managed by different administrative domains in different VOs. The TSRB is coined "trusted", because (1) it is deployed in an administrative domain trusted by the data owner, and (2) it is trusted to enforce data-owner specified access control policies. The TSRB controls access to data items or collections by combining User-based Access Control Lists (*User ACLs*) with required *host properties* that must be met by a remote host before the data can be accessed by a job on this host. Required host properties are described by the data owner in a *Remote Host Property List* (RHPL). Each host has a *Host Property List* (HPL) that contains host configuration details. The HPL contents are matched with the data's RHPL at connection time. The HPL is maintained by the remote host (Cluster A in Fig. 2), and is signed by the host's administrator. The TSRB also maintains for each data item or collection a *Host ACL* containing a list of administrative domains or hosts, who are trusted by the data owner for confidentiality and to *provide correct information* in their HPL.

The main actions are illustrated in Fig. 2. A user uploads data to the TSRB, e.g., using gridFTP (step 1). The data is stored in a storage system maintained in the TSRB domain. Metadata can be stored in a separate service managed by the TSRB, e.g., a File Catalog in case of storing files (step 2). A job is submitted through an external CRB (step 3), about which the data owner has no information. Eventually, the CRB submits the job to a cluster (step 4) that must be trusted by the data owner before the job can access data. As part of the protocol before data access is authorized, user (job) and host authentication takes place, and the data's RHPL and the remote host's HPL are compared (details are given in section 6.2). If RHPL and HPL match, a *microcontract* is established, which is a statement containing agreed-upon host properties and signed by both the TSRB and the remote host. Microcontracts are established for all authorization decisions, including, e.g., resolving file names in a File Catalog (step 5), and accessing the data item itself (step 6). Only after the TSRB receives a microcontract, are the data shipped to the job or middleware acting on the job's behalf. In step 7 a job returns to its CRB where it can be collected by its owner. Subject to agreement in the microcontract, Cluster A ensures that no data from the job's execution remains on the host.

Auditing is important to allow data owners to track which jobs applied which operations on their data, on behalf of which users, and from which hosts. All established microcontracts are shipped to an auditor process (see Fig. 2), which can be used by data owners to trace the transactions. Auditing can help establish trust (e.g., using reputation-based mechanisms), and enables tracking of potential sources of information leakage.

6.1 Concepts and Interactions

Job Authentication. One solution to provide a secure binding between jobs and Proxy certificates is to combine job integrity verification with a trust-based mechanism. Only if a data owner trusts a remote system to verify the integrity of incoming jobs properly, can he or she assume the the job-Proxy certificate binding to be valid, and can Proxy certificate-based authentication be trusted. Job integrity verification can be implemented securely if all initial content of the job is signed by its owner, thus creating an unforgeable binding between all components of a job. For example, a secure *job container* could be created before submitting the job, which should be signed using the owner's private key - see a similar idea in [24]. A job container has a well-defined structure, which makes it straightforward for the middleware to find the components of the job that are relevant for integrity verification. Other implementations are also conceivable, e.g., using signed Virtual Machine images [23].

Host Property Lists. For risk assessment and policy enforcement, hosts should announce security relevant properties of their operating system, its configuration, and the used middleware, including properties regarding job integrity verification, in their Host Property List (HPL). The host administrator has the responsibility to fill in the HPL correctly. As a concrete example, the HPL could report on whether the operating system was configured to use encrypted swap space, and on whether the middleware is capable of job integrity verification and provides jobs with a private file system that is removed after the job exits.

HPLs allow for run-time assessment on whether a host adheres to the requirements for secure data handling as imposed by a data owner. This assessment takes place at the time that a connection is made to the TSRB. No trusted third party is required for storing and/or matching properties in a host's HPL, allowing the system to scale.

Microcontracts state the obligations that the site holds with regard to a transaction. Our framework requires that all Grid middleware components that are concerned with data transfer aspects (e.g., gridFTP) are extended with functionality to report a signed HPL to their peer processes at connection time. Based on whether peers trust each other to provide correct information, and on the information in their HPLs, both parties decide whether to proceed with the transaction (e.g., data transfer), which takes place over a mutually authenticated secure channel. Agreement should be reached on the properties in the data item's RHPL *before* any data is shipped. For *non-repudiation*, both parties must co-sign a *microcontract* once agreement is reached. Non-repudiation means that none of the parties can deny that they agreed on the contract's content. To allow for auditing the exact operations on a particular data item, the microcontract has to be bound to each individual transaction, by including e.g., a hash over the data and the operation.

Trusted Storage Resource Broker. The TSRB is the key component for managing all privacy sensitive data in our framework. The TSRB is the central *reference monitor* and access point for data stored through this TSRB. In particular, the TSRB enforces the access control policies outlined in this paper. For clarity of exposition, we assume that the TSRB is a non-distributed service running in a single domain. The TSRB (and by implication, domain) is determined as trusted by a data owner prior to storing data on it. Although we refer to the TSRB as a *resource broker* here, the TSRB is effectively an abstraction for a secure storage system. In case where the TSRB uses distributed facilities (e.g., untrusted storage elements managed by different domains), the TSRB can implement broker functionality. In this case, the TSRB should make sure that it stores only encrypted data on untrusted storage, using cryptographic filenames. Example storage systems that are implemented as a broker for encrypted data are described in [20, 26].

Naming and Metadata Services. The TSRB can offer metadata services for managing and querying metadata about the stored data. Metadata is useful to search for data items of interest in large data collections. File names can be seen as metadata specific for file systems. Naming or metadata services must be integrated into the TSRB, since access to file names and other sensitive metadata should be carefully protected. For example, careless encoding of file names could enable attackers to identify patient or hospital information from a file name and re-identify a patient. Naming or metadata services may be private to a VO, or part of some hierarchical naming service. In either case, file name lookup requests are subject to data-owner specified access control policies as outlined in this paper.

Access Control Lists. Access control in our system is enforced on the basis of ACLs. ACLs can be associated with individual data items or with a grouping (set) of data items. In case of files, grouping may be facilitated by e.g., associating ACLs with directory names. Unauthorized users should not even be able to find out if a given data item exists.

The *User ACL* contains a list of principals (job owners) that are allowed to access a (set of) data item(s), together with these principals' access rights on that data. The Host ACL specifies from what hosts or domains authenticated jobs may access particular data, and with what access rights. Access rights from the User and Host ACLs are combined such that only the minimum set of rights for this data is granted to a job of a given user running on a given host.

The trusted domains or hosts in the *Host ACL* are determined by the data owner, e.g., based on whether he or she trusts the administrator of a particular administrative domain. Host ACLs are expressed as GSI host/domain name patterns, which match with the common name field of the x509 GSI host certificate, e.g., *.sara.nl, or host1.amc.nl. Specific patterns override wildcarded patterns.

Also associated with data items or sets of data is a *Remote Host Property List (RHPL)*. Before evaluating a remote host's HPL, it is checked that this host is in the Host ACL; only then is the HPL information considered trusted. We chose to separately store an RHPL with each (set of) data items, in addition to the basic User and Host ACLs, because of the dynamic nature of Grid systems. Different domains may contain many machines or clusters, each of which with different configuration and job or data handling properties, which may even change over time. Connection-time RHPL / HPL matching allows the system to evaluate these properties at runtime.

Job Submission Procedure. At job submission time, a host must be selected from which the job's input data is accessible. Since CRBs are generally not trusted, client-side software should be used which contacts the TSRB before job submission. Client-side software can authenticate directly to the TSRB using the owner's identity key. If authorized to access the data, it can fetch the relevant access control information, using which a job description is created. To allow for selection of suitable hosts by the CRB, HPLs could be published in a (global) information system. Note that because of run-time (R)HPL evaluation, the information system does not need to be completely consistent or trusted. For this reason, we believe the approach is scalable.

Auditing is important to allow for tracing all operations on a particular data item. For convenience and scalability, we use a trusted *auditor process* per TSRB, managed by the TSRB. Copies of the co-signed microcontracts of all transactions are sent to the auditor. This allows the data owners to trace all transactions that involve a particular data item securely, i.e., in a way that ensures non-repudiability.

6.2 Putting it All Together

Authorization of a data access requires that the connecting job's owner is on the User ACL, that the host on which the connecting job runs is on the Host ACL, and that the properties in the RHPL match the properties in the connecting host's HPL. Authorization of a data request consists of the following steps, assuming GSI host/Proxy certificate based authentication. **1.** At connection time, the connecting process (either a job or middleware, in case of data pre-fetching) authenticates with the TSRB using the job's Proxy certificate, resulting in an authenticated and encrypted SSL/TLS channel. **2.** The information from the Proxy certificate is matched against the User ACL to see if access is allowed. If not, an error is returned that does not indicate whether the data exists. **3.** The TSRB and the connecting process engage in a protocol for matching RHPL and HPL properties. If the connecting process is the middleware (e.g., during data pre-fetch), it can directly sign the microcontract. If the connecting process is a job, it has to request its local middleware (using a runtime interface) to match the RHPL of the TSRB with the host's HPL, and to have it sign a microcontract on its behalf if these properties match. The microcontract includes the (hash over the public key of the) Proxy certificate of the job to which it was issued. **4.** The signature over the microcontract (shipped together with the GSI host certificate that was used for signing) is compared with the Host ACL, to see if the HPL information is trusted and if access is allowed from this host.

The above mechanisms suffice to establish the required combination of Host ACL and User ACL based authorization, together with obtaining a microcontract signed by the connecting host before the data is shipped. If all provided information matches the data owner's requirements, the data is shipped to the requesting job or middleware, and the microcontract is logged in the auditor process.

7 Conclusion

We presented a trust-based security framework for Grid middleware that allows for enforcement of access control and data export policies for privacy-sensitive data. The framework proposes a Trusted SRB to manage data and enforce fine-grained access control policies on behalf of data owners. Access control policies combine user-based access control and trusted hosts lists, with a runtime evaluation of properties of remote hosts from which jobs request data access. Microcontracts allow for establishing data handling agreements, and a secure auditing mechanism based on microcontracts allows for tracing all operations on the data. We also outlined an approach for secure job authentication in the context of the GSI security infrastructure.

The focus of our paper is on usage scenarios where Grid-based storage and data sharing is required. Our framework emphasizes *data-owner specified* user and host (property) based access control policies, to ensure that privacy sensitive information is only made accessible to authorized jobs running on hosts trusted by the data owner, which meet the data owner's requirements for secure data handling.

Acknowledgments

We thank Oscar Koeroo, Dennis van Dok, and David Groep (NIKHEF) for valuable insight in the gLite-based VL-e infrastructure, as well as Keith Cover (VU Medical Center) for information on his job farming application.

References

[1] dcache storage system. http://www.dcache.org/.
[2] European commission, data protection regulations overview page. http://ec.europa.eu/justice_home/fsj/privacy/.
[3] glite (grid middleware). http://glite.web.cern.ch/glite.
[4] Globus alliance toolkit homepage. http://www.globus.org/toolkit/.
[5] Job submission description language (jsdl) specification, version 1.0. http://www.gridforum.org/documents/GFD.56.pdf.
[6] Lhc computing grid project. http://lcg.web.cern.ch/LCG.
[7] Special publication 800-88: Guidelines for media sanitization by the national institute of standards and technology (nist). http://csrc.nist.gov/publications/nistpubs/#sp800-88.
[8] U.s. safe harbor framework. http://www.export.gov/safeharbor/.
[9] Dutch ministry of health, welfare and sport - wet op de geneeskundige behandelingsovereenkomst (wgbo), 1994. http://www.hulpgids.nl/wetten/wgbo.htm.
[10] Dutch ministry of health, welfare and sport - wet medisch-wetenschappelijk onderzoek met mensen (wmo), 1998. http://www.healthlaw.nl/wmo.html.
[11] R. Alfieri et al. Voms, an authorization system for virtual organizations. In *European Across Grids Conf.*, volume 2970 of *LNCS*, pages 33–40. Springer, 2004.
[12] U. Congress. Health insurance portability and accountability act, 1996.
[13] S. Erberich et al. Globus medicus - federation of dicom medical imaging devices into healthcare grids. *Studies in Health Technology and Informatics*, 126:269–278, 2007.
[14] S. Fischer-Huebner. *IT-security and privacy: design and use of privacy-enhancing security mechanisms*. Springer-Verlag, New York, NY, USA, 2001.
[15] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational grids. *5th ACM Conf. on Computer and Communication Security*, pages 83–92, November 2-5 1998.
[16] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *Int'l J. Supercomputer Applications*, 15(3), 2001.
[17] J. Herveg. The ban on processing medical data in european law: Consent and alternative solutions to legitimate processing of medical data in healthgrid. In *Proc. Healthgrid 2006*, volume 120, pages 107–116, Amsterdam, The Netherlands, 2006. IOS Press.
[18] D. Kalra et al. Security and confidentiality approach for the clinical e-science framework (clef). *Methods of Information in Medicine*, 44(2):193–197, Feb. 2005.
[19] B. Malin. Compromising privacy with trail re-identification: The reidit algorithms. *CMU Technical Report, CMU-CALD-02-108*, December 2002. Pittsburgh.
[20] J. Montagnat et al. A secure grid medical data manager interfaced to the glite middleware. *J. of Grid Computing*, 2007.
[21] S. D. Olabarriaga et al. Towards a virtual laboratory for fmri data management and analysis. In *Proc. HealthGrid 2006*, volume 120, pages 43–54, Amsterdam, The Netherlands, 2006. IOS Press.
[22] L. Sweeney. k-anonymity: A model for protecting privacy. *Int'l J. of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.
[23] F. Travostino et al. Seamless live migration of virtual machines over the man/wan. *Fut. Gen. Comp. Systems*, 22(8):901–907, 2006.
[24] G. van 't Noordende, F. Brazier, and A. Tanenbaum. Security in a mobile agent system. *1st IEEE Symp. on Multi-Agent Security and Survivability*, Aug. 2004. Philadelphia.
[25] G. van 't Noordende et al. A secure jailing system for confining untrusted applications. *2nd Int'l Conf. on Security and Cryptography (SECRYPT)*, pages 414–423, July 28-31 2007. Barcelona, Spain.
[26] L. Xu. Hydra: A platform for survivable and secure data storage systems. *ACM StorageSS*, November 11 2005.