Security Requirements Analysis for Large-Scale Distributed File Systems*

Syed Naqvi[1], Olivier Poitou[1], Philippe Massonet[1], and Alvaro Arenas[2]

[1] Centre of Excellence in Information and Communication Technologies (CETIC), Belgium
{syed.naqvi,olivier.poitou,philippe.massonet}@cetic.be
[2] CCLRC Rutherford Appleton Laboratory, UK
a.e.arenas@rl.ac.uk

Abstract. This paper presents an analysis of security requirements of large-scale distributed file systems. Our objective is to identify their generic as well as specific security requirements and to propose potential solutions that can be employed to address these requirements. *FileStamp* – a multi-writer distributed file system developed at CETIC is considered as a case study for this analysis. This analysis yields that the existing range of security solutions can be employed to secure large-scale distributed file systems. However, they should be holistically employed to triumph over the security chinks in the *FileStamp*'s armor.

Keywords: security services, requirements analysis, highly scalable systems, distributed data management.

1 Introduction

The exponential growth in the scale of distributed data management systems and corresponding increase in the amount of data being handled by these systems require efficient management of files by maintaining consistency, ensuring security, fault tolerance and good performance in terms of availability and security. Read only systems such as CFS [1] are much easier to design as the time interval between meta-data updates is expected to be relatively high. This allows the extensive use of caching, since cached data is either seldom invalidated or kept until its expiry. Security in a read-only system is also quite simple to implement. Digitally signing a single root block with the administrator's private key and using one-way hash functions allow clients to verify the integrity and authenticity of all file system data. Finally, consistency is hardly a problem as only a single user, the administrator, can modify the file system.

Multi-writer file systems face a number of operational issues not found in the read only systems. These issues include maintaining consistency between replicas, enforcing access control, guaranteeing that update requests are authenticated and correctly processed, and dealing with conflicting updates.

* This research work is supported by the European Network of Excellence **CoreGRID** (project reference number 004265). The network aims at strengthening and advancing scientific and technological excellence in the area of Grid and Peer-to-Peer technologies. The CoreGRID webpage is located at www.coregrid.net

This paper is organized in the following manner: an overview of *FileStamp* distributed file system is presented in section 2. Its generic and specific security requirements are elaborated in section 3. Section 4 presents a detailed account of technologies that can be employed to address the security requirements of the *FileStamp*. Finally some conclusions are drawn in section 5.

2 *FileStamp* Architecture

FileStamp is a distributed file system developed at CETIC with the aim of finding a solution to the problems encountered in multi-writer file systems. It is a highly scalable, completely decentralized multi-writer peer-to-peer file system. The current version of the *FileStamp* is based on Pastis [2] architecture. It aims at making use of the aggregate storage capacity of hundreds of thousands of PCs connected to the Internet by means of a completely decentralized network. Replication allows persistent storage in spite of a highly transient node population, while cryptographic techniques ensure the authenticity and integrity of file system data.

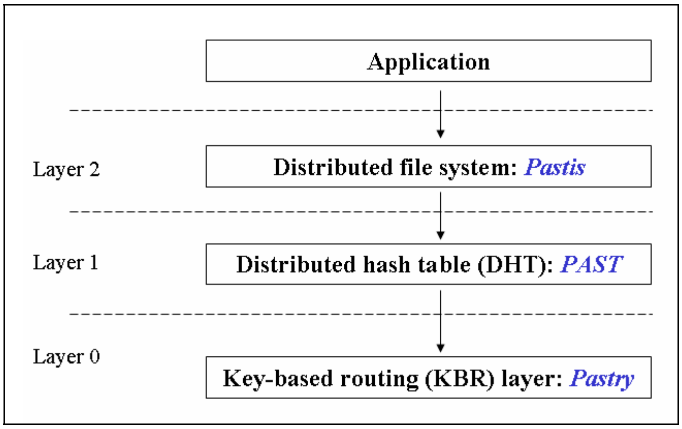
Application
Layer 2
Distributed file system: Pastis
Layer 1
Distributed hash table (DHT): PAST
Layer 0
Key-based routing (KBR) layer: Pastry

Fig. 1. *FileStamp* Layered Architecture

The layered architecture of *FileStamp* is shown in figure 1. Routing and data storage are handled by the Pastry [3] routing protocol and the PAST [4] distributed hash table (DHT). The good locality properties of Pastry/PAST allow Pastis to minimize network access latencies, thus achieving a good level of performance when using a relaxed consistency model. In Pastis, for a file system update to be valid, the user must provide a certificate signed by the file owner which proves that he has write access to that file.

The format of the Pastis certificate is shown in figure 2. This certificate is issued by the file owner and it grants the write access to a given user. The expiration date allows access revocation.

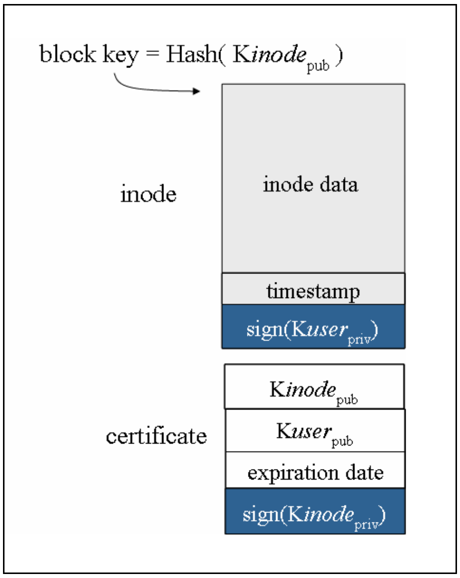

Fig. 2. Pastis certificate format

Authentication of the certificate is performed by the DHT nodes and FS clients. They verify both signatures when storing and/or retrieving a UCB (User Certificate Block).

This certificate has two crucial problems. First, it always gives *write* permission to its users whereas in a real life application, a user may only be given *read* permission while accessing the file. Second, its format is not standardized. It does not correspond with the format of the X.509 certificate and hence it renders compatibility problem with the existing standard credentials. This issue is discussed in detail in section 4.1

3 Security Requirements of *FileStamp*

The security requirements of *FileStamp* are driven by the roadmap of Open Grid Services Architecture (OGSA) [5]. OGSA security model casts security functions as OGSA services. This strategy allows well-defined protocols and interfaces to be defined for these services and permits an application to outsource security functionality by using a security service with a particular implementation to fit its current need.

3.1 Generic Requirements

This is the set of security services that constitutes the fundamental requirements for any data management system.

3.1.1 Authentication

Authentication provides plug points for multiple authentication mechanisms and the means for conveying the specific mechanism used in any given authentication operation. The authentication mechanism may be a custom authentication mechanism or an industry-standard technology. The authentication plug point must be agnostic to any specific authentication technology.

Authentication between two entities of *FileStamp* nodes means that each party establishes a level of trust in the identity of the other party. In practical use an authentication protocol sets up a secure communication channel between the authenticated parties, so that subsequent messages can be sent without repeated authentication steps, although it is possible to authenticate every message. The identity of an entity is typically some token or name that uniquely identifies the entity.

3.1.2 Authorization

Authorization allows for controlling access to grid resources based on authorization policies (i.e., who can access a resource, under what conditions) attached to each service. It also allows for service requestors to specify invocation policies (i.e. who does the client trust to provide the requested service). Authorization should accommodate various access control models and implementation.

In the grid environments, the virtual organisations (VOs) [6] introduce challenging management and policy issues, resulting from often complex relationships between local site policies and the goals of the VO with respect to access control, resource allocation, and so forth. In particular, authorization solutions are needed that can empower *FileStamp* to set policies concerning how resources assigned to the *community* are used without, however, compromising site policy requirements [7].

3.1.3 Availability

Availability of a requested data item is an important performance parameter. A well-known technique for improving availability in distributed systems is replication. If multiple copies of data exist on independent nodes, then the chances of at least one copy being accessible are increased. Aggregate data access performance will also tend to increase, and total network load will tend to decrease, if replicas and requests are reasonably distributed.

3.1.4 Confidentiality

Confidentiality is the property that information does not reach unauthorized individuals, entities, or processes. It is achievable by a mechanism for ensuring that only those entitled to see information or data have access to that information. The confidentiality requirement includes point-to-point transport as well as store-and-forward mechanisms.

3.1.5 Integrity

Integrity is the assurance that information can only be accessed or modified by those authorized to do so. Data integrity is a nontrivial problem especially when storage hardware and networks are not perfect. Data loss and corruption must be timely caught and swiftly fixed. As systems grow in size and complexity, problems may pass unnoticed until recovery becomes difficult and expensive.

3.2 Specific Requirements

This is the set of security services that are specifically needed for *FileStamp*. These services complement the generic set of security services and are needed to enhance the quality of security of the data management system.

3.2.1 Resilience

Resilience is an important requirement as the grid links and nodes are very dynamic in nature and may change over the time. *FileStamp* security architecture should remain intact and should deliver the promised level of security assurances even if its composition changes over the time. The resilience provides an abstraction layer to hide the architectural changes from the overall security architecture.

3.2.2 Data Lifecycle Management (DLM)

Data Lifecycle Management (DLM) is the process of managing data throughout its lifecycle from conception until disposal across different storage media, within the constraints of the entire process. The lifecycle is the time from the moment data is created until it is deleted or stored indefinitely. Security assurances require spanning the entire lifecycle of data. *FileStamp* should ensure that the data contents will be protected from the malevolent entities throughout its lifecycle.

3.2.3 Fault Tolerance

Fault tolerance is a desirable feature especially when transfers of large data files occur. Protocols such as GridFTP [8] allow for *resuming* transfers from the last byte acknowledged. Overlay networks provide *caching* of transfers via store-and-forward protocols. However, caching reduces performance of the overall data transfer and the amount of data that can be cached is dependent on the storage policies at the intermediate network points.

4 Solutions for the *FileStamp* Security Requirements

In this section, solutions to the security requirements of *FileStamp* are provided. The premier objective of this section is to identify the range of existing technologies that can be employed in *FileStamp*. However, solutions to all the security requirements do not already exist. In situations where existing solutions are either inadequate or non-existent, we have discussed the potential solutions and have given reference to our ongoing work in that direction.

The aim of this approach is to workout new solutions which are needed for the security architecture of the grid data management systems without reinventing the wheel.

4.1 Authentication

Most of the current grid tools are built on Grid security Infrastructure (GSI) [9] or Secure Hyper Text Transfer Protocol (HTTPS) [10], both of which use X.509 certificates [11] for securely establishing a grid identity [12].

Other schemes include PGP keys [13], SSH keys [14], and SPKI [15] keys and protocols. SPKI focuses on authorization certificates more than identity certificates. SSH is primarily a private/public key mapping with no real attempt to provide global names. The X.509 scheme has a small set of trusted third parties called Certification Authorities (CAs). These CAs are used to sign identity certificates that contain subscriber's public key. This improves the scaling properties of public key distribution in that only the CA's public key needs to be distributed in an out-of-band secure manner. In systems without a trusted third party, such as PGP, each key holder must find some secure way of establishing the association of his identity with his public key, to each party with which he wishes to establish authenticated communication. In the X.509 infrastructure, the individual subscriber's public key can be transmitted in a public key certificate as part of a TLS connection handshake and can be accepted as valid if the certificate is signed by a trusted CA. Another feature of the X.509 infrastructure is that it supports multiple independent CAs. In a Grid each site may chose which CAs it will accept for binding domain names and public keys.

We recommend the use of X.509 infrastructure for *FileStamp*. It will not only standardize its authentication mechanism (unlike owner's issued certificates) but also facilitate its interactions with the grid world. *FileStamp* with X.509 infrastructure will be easily integrated with any grid platform. Initially, a local CA can be created that will deliver the standard X.509 certificates to the bona fide users of *FileStamp*. Later the certificates of other CAs (such as Belgian Grid CA [16]) can be used for authentication purposes.

4.2 Authorization

FileStamp may simply employ local mapping of the users (like UNIX authorization matrix). This mapping also serves as an access control check – access to the resource is denied if the user is not listed in the local mapping configuration. In this scheme, once the user is mapped to a local identity, local policy management and enforcement mechanisms constrain the user's actions to those allowed by local policy. This approach allows the local operating system to act as a sandbox. Thus, administrators can use normal policy administration tools to configure policy.

This simple approach has the advantage of being easy for site administrators to understand and configure because it uses existing local policy management and enforcement mechanisms with which the administrator is presumably already familiar. However, in the context of the grid environment, this approach has several shortcomings (such as scalability, lack of expressiveness, consistency of policies, etc.).

These problems are addressed in the Community Authorization Service (CAS) [17]. The idea behind the evolution of CAS is inspired from the Role Based Access Control (RBAC) [18]. CAS allows for a separation of concerns between site policies and VO policies. Specifically, sites can delegate management of a subset of their policy space to the VO. CAS provides a fine-grained mechanism for a VO to manage these delegated policy spaces, allowing it to express and enforce expressive, consistent policies across resources spanning multiple independent policy domains. CAS implementations are built on the Globus [19], thus allowing for easy integration of CAS with existing Grid deployments.

Other solutions include VOMS [20], Akenti [21], and PERMIS [22]. VOMS (Virtual Organization Management Service) and CAS are similar architecturally in that both issue policy assertions to a user that the user then presents to a resource for the purpose of obtaining VO issued rights. The primary difference between the two systems is the level of granularity at which they operate. The policy about what memberships a user has is centralized in the VOMS server, but the policy regarding exactly what rights those memberships grant is distributed among the sites. CAS assertions provide the rights directly and do not need interpretation by the resource. This complete centralization of policy can achieve better consistency especially in situations where policies are changing dynamically.

Akenti and PERMIS, while having differences in implementation and features, are architecturally similar in that they provide a resource with an authorization decision in regards to a request. While the CAS implementations provide simple authorization decision functionality, they are limited to supporting CAS policy assertions and do not have as rich a feature set as either Akenti or PERMIS. It is possible that either of these systems, with some modifications, could be used to provide resource-side functionality for CAS (i.e., parse the CAS assertion and use it to authorize the user's request.)

We recommend the use of CAS with the implementation of a local authorization server for *FileStamp*. Local authorization server would accept authorization queries from request servers, apply all applicable local and community policies, and return a yes or no answer. This authorization server would need to be highly trusted by the resource server and highly available. This service could potentially take CAS credentials, forwarded by the resource, and use their credentials in making its decision, or it could contact the CAS server itself. Such a server could be implemented by using Akenti or PERMIS.

4.3 Availability, Confidentiality, and Integrity

Grid technologies enable transparent access to a wider resource pool, across organizations as well as within organizations; they can be used as a building block to realize stable, highly reliable execution environments. In such a complex environment, policy-based autonomous control and dynamic mobility are keys to realizing systems that are highly flexible and recoverable. Availability is often not considered in literature, when it comes to a model design. Nevertheless, in a production environment we cannot expect user not having assurances regarding the availability of what they pay for. GSI provides mechanisms to grant availability of data owned by a user on a remote resource. These are achieved by means of secure communication protocols, such as HTTPS. As far as services availability is concerned, Globus relies on a dedicated module that manages a limited set of grid events.

Use of some adequate encryption technologies is indispensable to guarantee the secure communications across the grid nodes which assure the confidentiality and also integrity. Encryption indirectly assures the availability too; however, the protection against the denial of service attack is addressed in the security policy. There exist a range of encryption technologies from HTTPS (where a layer of security is added on the top of HTTP) to Secure Hash Algorithm (SHA) [23] (where it is computationally difficult or impossible to hack and the integrity check – checksum – is also performed).

Figure 1 shows graphic representations of these two encryption schemes. In figure 1a, the layered architecture of HTTPS is shown. Figure 1b depicts how the SHA works. The quick comparison of these two techniques show that SHA seems quite powerful as it require considerable computing power to break the algorithm; however, in the specific context of the grid applications notably *FileStamp*, we need to consider the overhead incurred due to the encryption operations. Large datasets will consume enormous

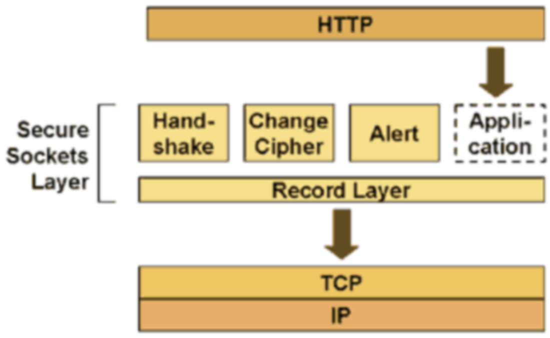

Fig. 1a. HTTPS Architecture

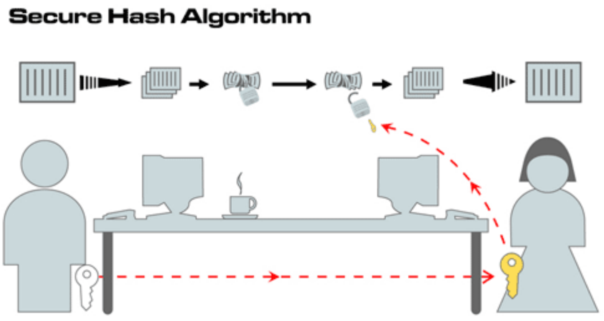

Fig. 1b. SHA Architecture

computing cycles for the SHA processing and HTTPS may not be considered as dependable solution especially when network connections are not reliable.

We recommend the use of encryption technology for *FileStamp* as the data movements across the grid nodes will be subject to potential attacks if there will be plain text data exchange between the nodes. However, the selection of some specific encryption technology is a tricky issue that depends on the nature of data (required security level of the data movement) and the affordability of the total cost of the encryption algorithms. A simple technique such as HTTPS can be employed for generic situations and some more powerful techniques can be used for providing higher level of security assurances. SHA consumes enormous amount of computing power but in return it provides highest security assurances.

4.4 Resilience and Fault Tolerance

General trend for the attainment of resilience and fault tolerance in the distributed systems is to maintain ample number of replicas of the dataset. When some node fails then the load/job is transferred to some other node. The quality of service depends on how efficiently the system recognizes the faulty nodes and how transparently the jobs are migrated from the faulty nodes to working nodes without interrupting operations.

In order to assure resilience and fault tolerance features, *FileStamp* should be able to negotiate the terms of security parameters with the nodes so that new replicas be created if the set of nodes expands resulting in the need of more replicas; or failure of some existing nodes bearing replica sets need to be compensated by generating new replicas.

We recommend the phased approach (as mentioned in [24]) to deal with the resilience and fault tolerance issue. According to this approach:

1. In Phase I, the service providers that need to interact are identified. It is generally assumed that this is undertaken through a manager entity – which is forming the VO in order to undertake a particular activity.
2. In Phase II, the identified providers are asked to join the VO. This phase may involve negotiation between the manager entity and the providers (or directly between the providers) to ensure that a Service Level Agreement (SLA) is established between the entity and each provider (or directly between the providers).
3. In Phase III, the providers interact to perform the particular activity desired by the manager entity.

A set of protocols is needed to perform these negotiations. Negotiation protocols are the set of rules that govern the interaction. They are required to realize SLA-aware resource management system.

We recommend the use of *Service Negotiation and Acquisition Protocol (SNAP)* [25] as negotiations protocol. SNAP is structured around the negotiation of SLAs to solve the negotiation problems at run-time. When SNAP is used to submit a file transfer job to a community scheduler, the scheduler understands that a transfer requires substantial storage space on the destination resource, and substantial network and endpoint I/O bandwidth during the transfer. The distributed applications (common in Grid environments) exacerbate the coordination problems of community schedulers. Not only do SLAs coordinate use of resources by mutually distrustful schedulers, they also coordinate the use of distrustful resources for a single application goal. The file transfer emphasizes such distributed goals by requiring real-time coordination of significant endpoint and network capability.

4.5 Data Lifecycle Management (DLM)

Data lifecycle management (DLM) is a policy-based approach to managing the flow of an information system's data throughout its life cycle – i.e. from creation and initial storage to the time when it becomes obsolete and is deleted. Security assurances require spanning the entire lifecycle of data. Existing Grids are already managing huge quantities of data [26]. Since Grids maximize the utilization of computing resources, their potential to generate new data and consume storage is very high, making storage capacity and DLM critical issues. By targeting data to appropriate storage media (primary disk storage, secondary serial advanced technology attachment (ATA) storage, tape, etc.) DLM solutions can influence on the overall protection of the data besides significantly reducing the cost of Grid storage infrastructures. *FileStamp* should ensure that the data contents will be protected from the malevolent entities throughout its lifecycle.

We recommend a two-tier approach to handle the DLM issue in the *FileStamp* system:

First, the security policy should explicitly mention the desired lifecycle of the data being managed by the *FileStamp* system. The dynamic nature of the grid environments does not permit some rigid definition of any parameter including security; however, the security policy of a VO is generally fixed for that VO and hence the VOs using the *FileStamp* should include a formal description of the stage where the data generated by the VO operations be destroyed from the storage devices.

Second, FileStamp should also employ some secure storage management technique such as HSM (Hierarchical Storage Management) [27]. HSM is policy-based management of file backup and archiving in a way that uses storage devices economically and without the user needing to be aware of when files are being retrieved from backup storage media. The hierarchy represents different types of storage media, such as redundant array of independent disks systems, optical storage, or tape, each type representing a different level of cost and speed of retrieval when access is needed.

5 Conclusions

Global connectivity of computing and storage resources opens up the possibility of misusing information to a degree never seen before. The objective to facilitate use of these resources by protecting them against any misuse must, however, be realistic given the current technical infrastructure. It is important that the security technologies be integrated in these systems from the inception stage rather than considering them as add-on optional features. Security issues should not be overlooked while designing these systems as they are critical to the success of these scalable distributed systems.

In this paper, the security requirements of large-scale distributed file systems are addressed. The *FileStamp* multi-writer distributed file system is considered as a case study for this analysis. Various security requirements are identified and the potential solutions corresponding to these requirements are proposed. However, it is important to remember that the analysis of security requirements is a process, the risk and threat pictures are always changing, and their analysis needs to be continuously updated. In other words, overall infrastructure of large-scale distributed file systems should be subject to constant review and upgrade, so that any security loophole can be plugged as soon as it is discovered.

References

1. Dabek F., Kaashoek M., Karger D., Morris R., and Stoica I., *Wide-Area Cooperative Storage with CFS*, In the proceedings of 18th ACM Symposium on Operating Systems Principles (SOSP'01), chateau Lake Louise, Banff, Canada, October 2001
2. INRIA Project PASTIS http://regal.lip6.fr/projects/pastis/pastis_fr.html
3. Rowstron A. and Druschel P., *Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems*, Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), 2001, pp 329-350

4. Druschel P., and Rowstron A., *Past: Persistent and Anonymous Storage in a Peer-to-Peer Networking Environment*, Proceedings of the 8th IEEE Workshop on Hot Topics in Operating Systems (HotOS-VIII)? 2001), pp. 65-70
5. Welch V., Siebenlist F., Foster I., Bresnahan J., Czajkowski K., Gawor J., Kesselman C., Meder S., Pearlman L., Tuecke S., *Security for Grid Services*, Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03), 2003
6. Foster I., Kesselman C., and Tuecke S., *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*. International Journal of High Performance Computing Application, 15 (3), pp. 200-222, 2001
7. Foster I., Kesselman C., Pearlman L., Tuecke S., and Welch V., *The Community Authorization Service: Status and Future*, In Proceedings of Computing in High Energy Physics 03 (CHEP '03), La Jolla, California, USA, March 24-28, 2003
8. Allcock W. et al., *GridFTP: Protocol extensions to FTP for the Grid*, GGF Document Series GFD.20, April 2003
9. Foster I., Kesselman C., Tsudik G., Tuecke S., *A Security Architecture for Computational Grids*, ACM Conference Proceedings 1998, ISBN 1-58113-007-4, pp 83-92
10. Rescorla E., *Hyper Text Transfer Protocol (HTTP) over Transport Layer Security (TLS)*, Internet Engineering Task Force (IETF) draft RFC # 2818, May 2000
11. Chokhani S., *Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework*, Internet Engineering Task Force (IETF) draft RFC # 2527, March 1999
12. Thompson M., Olson D., Cowles R., Mullen S., Helm M., *CA-based Trust Issues for Grid Authentication and Identity Delegation*, Global Grid Forum (GGF) Certification Authority Operations Working Group Community Practices Document, Oct 2002
13. Garfinkel S., *PGP: Pretty Good Privacy,* O'Reilly & Associates, 1994
14. Barret D. and Silverman R., *SSH: The Secure Shell,* O'Reilly & Associates, 2001
15. Ellison C., *SPKI Requirements*, IETF RFC 2692 1999, http://www.ietf.org/rfc/rfc2692.txt
16. The Certification Authority of Belgian Grid Initiative – www.begrid.be/certification.htm
17. Pearlman L., Welch V., Foster I., Kesselman C., Tuecke S., *A Community Authorization Service for Group Collaboration.*, Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, 2002
18. Ferraiolo D., Cugini J., and Kuhn D., *Role Based Access Control (RBAC): Features and Motivations*, Proceedings of the 11th Computer Security Applications Conference, pp 241-248, New Orleans, LA, USA, 11-15 December 1995
19. Foster I. and Kesselman C., *Globus: A Metacomputing Infrastructure Toolkit*, International Journal of Supercomputer Applications, 11 (2). 115-129. 1998
20. VOMS Architecture v1.1, http://gridauth.infn.it/docs/VOMS-v1_1.pdf, May 2002.
21. Thompson M., Johnston W., Mudumbai S., Hoo G., Jackson K., and Essiari A., *Certificate-based Access Control for Widely Distributed Resources*, 8th Usenix Security Symposium, 1999
22. Chadwick D. and Otenko A., *The PERMIS X.509 Role Based Privilege Management Infrastructure*, 7th ACM Symposium on Access Control Models and Technologies, 2002
23. National Institute of Standards and Technology, *Secure Hash Standard*, Federal Information Processing Standards Publication 180-1, April 17, 1995
24. Olmedilla D., Rana O., Matthews B., and Nejdl W., *Security and Trust Issues in Semantic Grids*, Proceedings of Schloss Dagstuhl Seminar no. 05271: Semantic Grid: The Convergence of Technologies, Dagstuhl, Germany, July 03-08, 2005

25. Czajkowski K., Foster I., Kesselman C., Sander V., Tuecke S., *SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems*, Lecture Notes In Computer Science; Vol. 2537, Revised Papers from the 8th International Workshop on Job Scheduling Strategies for Parallel Processing, pp 153-183, ISBN:3-540-00172-7, 2002
26. Silicon Graphics Incorporate (SGI), *SGI and Intel on the Grid – Unique Capabilities for Grid Computing*, Whitepaper, 2005
27. Watson, R., *High Performance Storage System Scalability: Architecture, Implementation and Experience*, Proceedings of 22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies 2005, pp145-159, 11-14 April 2005