

# SECURE LOGISTICAL NETWORKING IN VIRTUAL ORGANIZATIONS

Lukáš Hejtmánek, Luděk Matyska, and Michal Procházka

CESNET, z.s.p.o.,  
Institute of Computer Science,  
Masaryk University  
Czech Republic  
e-mail: xhejtman@mail.muni.cz, ludek@ics.muni.cz, michalp@ics.muni.cz

## Abstract

In this paper, we propose an architecture that extends Logistical Networking to use Grid authentication and authorization services. Our architecture guarantees that user is authenticated to all services included in network storage stack, the authorization granularity is also at the service level and all authorizations can be revoked at any moment by service providers. We also support access policies. These can limit maximum amount of distributed storage space allocated to a user or group of users or they can limit the maximum amount of time the client can keep his data within the distributed storage. Advanced access control to files is supported, administrators can define access conditions. The prototype implementation has been used to evaluate overhead associated with the security enhancements. If only capabilities are encrypted, the `COPY` command has a notable but constant overhead of 10 ms, all the other basic commands experience no visible overhead. When the full data encryption is enforced, all the data manipulation commands are bound by the speed of the used AES 128 cipher.

## 1 Introduction

The demands on the capacity, availability, and performance of data storage are ever increasing and the requirements cannot be served with a centralized approach. Distribution of data storage is becoming very important aspect of data processing in many applications. Grids are an example of a distributed computing and data environment, where long term storage and retrieval of data in a distributed matter is the focus of interest, as demonstrated by many projects dealing e.g., with the high energy physics data (DataGrid, DataTag, EGEE, LCG, etc.).

Within Grids, users are usually organized within Virtual Organizations (VO) [4]. The VO serves as a unit for user membership and resource management, including the data management. Different approaches to data management are employed in current Grids, like Storage Elements which communicate with the computing nodes in a stage-in/stage-out manner, and networked and distributed file systems (NFS, AFS, GPFS, etc.).

The Logistical Networking [2] presents a novel approach to the problem of distributed data management usable in Grids. The Logistical Networking provides a universal storage fabric that provides a foundation for robust, scalable, and high perfor-

mance data storage systems that can be tailored to particular needs (e.g., both distributed file systems and storage elements could be built on such a fabric). The legacy Logistical Networking is not ready to be used in a Grid environment where proper authentication and authorization are necessary to control access to the stored data.

The goal of the work presented in this paper is to extend the Logistical Networking with the features needed for its incorporation into the VO-based Grid environment. We focus on the security aspects, including the authentication and authorization mechanisms compatible with the security infrastructure used by contemporary Grids. Our approach is based on the PKI infrastructure for authentication and VOMS service [1] for the authorization.

This paper is organized as follows: The Logistical Networking and VOs are briefly introduced in sections 2 and 3, respectively. The section 4 covers the abstract design of the proposed security enhancements to the Logistical Networking. Experimental results including performance evaluation are found in the section 5. The related work and conclusion are then covered by sections 6 and 7.

## 2 Logistical Networking

The Logistical Networking has been introduced to provide global network storage and data movement as a first class network service. Its major components are the IBP protocol and the corresponding network storage stack [2], consisting of five layers: the IBP, L-Bone, exNode, LoRS, and application layers.

The IBP layer represents the raw storage layer, storing individual data blocks without any knowledge about the actual data structure. Access to the stored data is controlled by system of capabilities. Only the user possessing appropriate capability is allowed to read or write data blocks. Capabilities are represented by the hostname of the IBP server and random strings generated by the IBP storage server. The capabilities are issued to the client while storing data block. To prevent indefinite storage allocation and resulting capacity exhaustion, only time-limited data storage is usually provided by the IBP servers. After expiration of the time-limit, the storage server may remove the data block at any time. Users can renew the “lease” of the storage capacity.

The L-Bone layer acts as a register of resources and gathers information about IBP storage servers such as hostname, location, and availability. The client connects the L-Bone server to get hostnames of the IBP storage servers with the data blocks and then communicates directly with the servers.

The exNode layer maps data blocks into files. This mapping—the file metadata—is represented by an XML file. The metadata contains capabilities to access particular data blocks. On the conceptual level, the metadata is equivalent to the well known UNIX I-node.

The LoRS layer comprises API for storing and retrieving data, and creating and using the metadata. Users can upload, download, and manipulate data on the IBP storage servers using command line utilities that are part of the LoRS layer.

The major drawback of usability of the LoRS API in a Grid environment lies in the metadata handling. The metadata is always sent to the client which is responsible for its storage. If the metadata is lost or corrupted, the original data is lost, too (at least, the

storage capacity is freed when the time limit expires). To overcome this problem, we have introduced a metadata manager that keeps the metadata on behalf of the client and ensures metadata availability. The metadata manager can be seen as an extension of the original network storage stack [5].

### 3 Virtual Organizations and Logistical Networking

A virtual organization [4] is a set of individuals and/or institutions that agree on sharing and access rules for some of their resources. The sharing is enabled by the appropriate policy and technical means to implement it. In this paper, we deal with sharing of storage resources based on the Logistical Networking. Each IBP depot has an owner with whom the particular VO negotiates the sharing rules. These rules must define sharing per VO, and the depot owner may also define additional per-user based sharing rules. The VO general rules are usually based on a VO membership. Both the membership and the per-user individualized rules require some kind of user identification, based on some authentication mechanism. In this paper, we use PKI [6] and VOMS (Virtual Organization Membership Service) [1] to provide the authentication and authorization, respectively.

Virtual organizations are usually not static entities, both resource providers and users can join and leave at any time. Also, a particular VO can become part of another VO or several VOs can merge together. The security infrastructure must be able to support even such dynamic behavior.

The IBP depots are registered with the L-Bone server. When dealing with Virtual organizations, the L-Bone sever should register only depots serving a particular VO. This restriction of resources registered with an L-Bone server can be seen as a natural enhancement of the VOMS server. Users, when connecting to the L-Bone, could easily see depots belonging to this VO, and the VO managers can easily open access to their IBP depot registering it to the appropriate VOs L-Bone server.

VOs are using some set of user names and group names for user authorization. These names are local and defined only within particular VO. This means that two distinct VOs can use the same user name or group name with different meaning. Consequently, services relying on these names cannot be directly shared among VOs, making merge of two or more VOs potentially difficult task. The basic authorization service within the Logistical Networking framework is provided by the metadata manager. When storing the metadata, we can either use implicit VO identification or we can include the VO identification, making it part of the metadata itself. In the former case, the metadata is local and the authorization information can not be shared among different VOs (as there may be name clashes). In the later case, the metadata is unique (assuming we have a globally unique VO identifier), so sharing the metadata information is rather easy. We decided to use this approach, as it can be usable even if no globally unique VO identifier exists—in such a case the merge of two VOs with the same identifier is not allowed (or, more precisely, we have to rename one of these VOs and then do the merge). This way, we keep the naming system local (within a VO) while allowing two or more VOs to merge without need for (expensive) renaming.

### **3.1 Data Manipulation Requirements**

The distributed storage system for a VO must fulfill the following requirements:

1. User must be authenticated to all services.
2. User must be authorized to use each service.
3. Authorization must be revocable.

Within the Logistical Networking context, this means that the user must be authenticated to all the metadata manager, L-Bone server, and IBP servers. This is achieved via a proxy certificate with appropriate VOMS attributes. The metadata manager must provide the metadata to the authorized users, the L-Bone server must issue a list of IBP servers to the authorized users, and the IBP servers serve the authorized users only. We require that each authorization must be eventually revocable (there is no permanent right to use some service). Instead of using revocation servers (or lists), we rely on time limited authentication and authorization information that must be periodically refreshed.

The access policy is usually defined on a group and user levels. The access policy defines maximum amount of storage space that can be consumed by the client and maximum amount of time for which the client can keep stored data. We also offer advanced access control to files. Administrators can define access conditions such as time limit to the file access (e.g., the condition can limit users or groups to access particular file only for one day), or system load limit (e.g., the condition can limit users or groups to access particular file only if storage server is idle). The owner of a file can bind the file with these usage constraints.

## **4 Architecture Design**

The proposed security architecture extension of the Logistical Networking guarantees proper user authentication and authorization to each element of the network storage stack. While users can cache the security related metadata to enhance the performance, all the authorization grants can be revoked, as only time-limited metadata are issued during the authorization process. The standalone services of the enhanced architecture do not need to contact any third party component to complete their tasks.

### **4.1 User identification**

The user identification can not be based directly on the certificates as defined and used by the PKI system. User may possess multiple certificates and even the certificates issued by a particular certificate authority can change in time, so the certificates are not stable enough to provide permanent user identification to the system. Therefore, we have introduced user and group ID attributes that are part of the X509v3 certificate extensions. The user and group IDs are defined to be unique and permanent and are signed as a part of the proxy certificate by a trusted authority (e.g., the VOMS server). When the user approaches the system for the first time, a new unique user ID is generated and associated with the submitted user certificate. Using this initial certificate, user is able to add multiple certificates, associating them all with the same user ID. Thereafter,

the user can provide any of his/her certificates to uniquely identify him/herself to the system.

## 4.2 Authorization Process

The authentication and the authorization has three parts:

1. The user obtains a *proxy certificate* with ID attributes (e.g., VOMS attributes or any other attributes defined within a particular VO). We suppose that the user has a certificate from a trusted certificate authority or a proxy certificate usable for this reason. The generation process of the proxy certificate with the VOMS attributes has two parts. First, the user creates a temporary proxy certificate from the personal certificate (or his proxy certificate), this temporary proxy certificate is used to authenticate with a VOMS server which issues VOMS attributes. Second, a new proxy certificate is created with the issued VOMS attributes. We suppose that the VOMS server issues the VOMS attributes only to authenticated and authorized clients, i.e., clients that are members of the particular VO. The proxy certificate has a limited time span that expires after couple of hours or days, thus making the authorization revocable.
2. The user obtains *metadata*. The metadata manager authorizes the user to access a file by issuing metadata specific to the file. Authorization is based on the ID attributes from the proxy certificate and on the access control list to the file. In this part, the authentication relies only on the ID attributes in the proxy certificate. Validity of the issued metadata is time-limited. We discuss this in Section 4.3.
3. The user obtains a *list of storage servers and signed permissions to use storage servers*. In case of data retrieval, the list of storage servers is not needed because the list of storage servers is included in metadata. The resource server issues the list of storage servers and the list of signed permissions to use storage servers to the authorized user (according to the ID attributes in proxy certificate). Validity of the issued signed permissions is also time-limited. The ID attributes from the user's proxy certificate are used while creating the signature, thus making it valid for the particular user only.

The user accessing a storage server must provide the particular piece of metadata and particular item (i.e., the signed permission to use particular depot) from the list of signed permissions to use storage servers. The storage server checks validity of the metadata and the signature of permissions. Thereafter, the client can access the stored data. In case of data storing, the user provides only particular item from the list of signed permissions.

We must enforce the users not to bypass any part of the authentication and authorization process, but we can see that this requirement is satisfied as the storage server requires authorization tokens from the register of resources and the metadata manager, and these two services require authorization tokens based on the users certificates issued by a trusted certificate authority. We also demand that the authentication and authorization tokens are revocable, which is provided by the time limited tokens that need to be refreshed and the refreshment may be rejected.

### 4.3 Time Limited Metadata

Splitting the metadata manager and storage servers into independent services poses security problem. If the metadata manager and storage servers are not online simultaneously, the storage server cannot verify the validity of metadata (i.e., whether access control information has not changed). The access control is not revocable as the client can keep issued metadata forever and thus has access to the data even if the access to metadata has been denied. However, revocable access control is one of the important features required by a VO. Therefore, the metadata manager must issue time limited metadata and the storage server must be able to verify validity of the metadata (i.e., the time limit). Basic idea about presenting time limited metadata is in signing metadata together with a time stamp by the metadata manager.

We build relation between a data block and the corresponding metadata. This relation can be kept at no extra cost. At the time of creation a new metadata file, the metadata manager generates unique short identification of the metadata file. As the client (the creator of the file) stores data into the storage server, it provides the identification of the metadata to the storage server. The storage server binds the identification and the data persistently. When the client is obtaining metadata from the metadata manager, the metadata manager signs only identification of metadata and time stamp (or some additional usage conditions, such as maximum load of depot) for the whole metadata. The client must provide this signature when accessing any data block from the storage server. The storage server verifies the signature, time stamp and binding data block and identification of metadata. Such signed data is of constant size. Unique identification can be done using identification of metadata manager (e.g., hostname) and increasing sequence starting from zero and being local to a particular metadata manager. Such identification is always unique if increasing sequence per hostname is maintained. Identification does not need to be changed when the file is renamed or moved into different metadata manager.

## 5 Preliminary Experiments

Our preliminary testbed consisted of a single client and a single server. The server has a dual Pentium Xeon@3.0 GHz processor, 4 GB RAM, 1 Gbps NIC and 7 TB SATA HW RAID 5 array, the client is 1.5 GHz Pentium-M, 768 MB RAM, 1 Gbps NIC with a local IDE disk. We are able to achieve 600 Mbps data transfer rate over a single TCP connection measured with the *iperf* tool.

We use simple client calling IBP functions to measure performance impact of the encryption support in the IBP protocol. The calling pattern is as follows: *Allocate* function, *Store* function, *Load* function, new allocation on the same server, and the *Copy* function. This pattern is repeated 400 times. The block size (i.e., the size of a data transfer per called function) has been 1 MB, 5 MB, and 50 MB. We compare legacy IBP implementation, the new version that encrypts only capabilities (Crypto), and the new version that encrypts both capabilities and transferred data (Full crypto).

The timing for the *Auth* operation (which performs the `IBP_auth` command) is presented in Table 1. As this operation is required once per established connection between the client and the server, its duration is constant and is independent of the

	<b>Auth</b>	<b>Allocate</b>
Legacy	–	$0.62 \pm 0.03$
Crypto, Full crypto	$14.5 \pm 1.6$	$1.10 \pm 0.04$

Table 1: Auth and Allocate duration in milliseconds.

size of data transfers. *Allocate* function causes penalty of about 0.5 ms, this penalty is independent of the size of the allocation and is independent of the data encryption (the use of full encryption does not have any influence on its duration). As we can see in the Table 2, 0.5 ms penalty is negligible compared to the duration of data transfers.

	<b>Block size</b>	<b>Load</b>	<b>Store</b>	<b>Copy</b>
Legacy	1 MB	$22.2 \pm 0.4$	$17.7 \pm 0.1$	$3.4 \pm 0.1$
Crypto	1 MB	$22.4 \pm 0.3$	$17.7 \pm 0.1$	$13.6 \pm 0.2$
Full crypto	1 MB	$87.4 \pm 0.6$	$92.6 \pm 0.3$	$84.2 \pm 0.4$
Legacy	5 MB	$111.6 \pm 0.8$	$86.7 \pm 0.1$	$11.8 \pm 0.4$
Crypto	5 MB	$113.1 \pm 1.5$	$86.6 \pm 0.2$	$21.6 \pm 0.3$
Full crypto	5 MB	$546.3 \pm 34.8$	$460.4 \pm 1.9$	$376.1 \pm 1.2$
Legacy	50 MB	$1106.8 \pm 3.0$	$862.9 \pm 19.1$	$120.3 \pm 27.6$
Crypto	50 MB	$1109.5 \pm 3.2$	$864.9 \pm 22.3$	$129.2 \pm 21.6$
Full crypto	50 MB	$4427.3 \pm 4.4$	$4569.0 \pm 6.3$	$3630.1 \pm 9.4$

Table 2: *Load*, *Store*, and *Copy* duration in milliseconds.

The timing for the *Load*, *Store*, and *Copy* operations are presented in Table 2. The basic *Load* and *Store* operations need comparable time to finish in the legacy and the new IBP version when only capabilities are encrypted. The new implementation of the *Copy* operations has a 10 ms penalty, caused by one additional round trip due to the certificate exchange, to the generation of a random key, and also to the preparation of the encryption layer. When the full encryption is used, the speed of the AES cipher (we use the `gcrypt` library) becomes the performance limiting step.

## 6 Related Work

The IBP protocol has been already extended to work together with the SSL to provide encrypted communication between user and the IBP storage depot. The OpenSSL implementation can be used through the LoRS command line utilities. However, SSL support is not used for authentication or authorization, and no interaction with the L-Bone layer occurs.

Authentication in the current grid environments is based on the same principle we use—the proxy certificates and set of trusted CAs [3]. However, we differ in the way authorization is performed. Common authorization technique used to access local data in a grid environment is based on a local decision on the resource [3]. A user obtains

authentication credentials (proxy certificate) and authorization credentials (VOMS attributes) and presents them to the resource. A resource has local mapping between the user identity in a proxy certificate and local UID/GID. Authorization is then based on local file system permissions. This approach cannot guarantee consistent view on the file permissions when files are distributed across several resources with different UID/GID assignments.

## 7 Conclusion

We have designed a system for distributed data storage based on the Logistical Networking with security based on certificates and VOMS attributes. This system is suitable for use within Grid VOs and it also supports services provided simultaneously to different VOs. The major feature of our framework is the added security layer, that guarantees that each user is properly authenticated to every element of the network storage infrastructure, that its authorization is checked at each stage, and also that each authorization must be potentially revocable. The security model also anticipates sharing of the storage fabric between different VOs.

We have designed and implemented a prototype implementation and performed preliminary performance evaluation. This evaluation shows that even the prototype implementation exhibits very favorable performance comparable to the legacy Logistical Networking storage stack.

## Acknowledgments

This research is supported by a research intent “Highly Parallel and Distributed Computing Systems” (MŠM 0021622419) and by the CESNET Development Fund project 172/2005. We would also like to thank to Petr Holub for stimulating discussions and help with the work described in this paper.

## References

1. R. Alfieri, R. Cecchini, V. Ciaschini, L. dell’Agnello, Á. Frohner, A. Gianoli, K. Lorentey, and F. Spataro. VOMS, an Authorization System for Virtual Organizations. In *Grid Computing, First European Across Grids Conference*, volume 2970/2004, pages 33–40. Springer Berlin / Heidelberg, 2004.
2. M. Beck, T. Moore, and J. S. Plank. An end-to-end approach to globally scalable network storage. *SIGCOMM Comput. Commun. Rev.*, 32(4):339–346, 2002.
3. EGEE: Site Access Control Architecture DJRA3.2. 2005.  
<https://edms.cern.ch/document/523948>.
4. Ian Foster, Carl Kesselman, and Steven Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Int. J. High Perform. Comput. Appl.*, 15(3):200–222, 2001.
5. Lukáš Hejtmánek. Distributed Data Storage with Data Versioning. In *CESNET Conference 2006*, pages 93–104. CESNET, z.s.p.o, 2006.
6. S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson. RFC3820: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile.  
<http://ietf.org/rfc/rfc3820.txt>.