Encrypted storage of medical data on a grid [*]

L. Seitz J.-M. Pierson
L. Brunie
LIRIS, CNRS FRC 2672, INSA de Lyon
7, av. Jean Capelle, 69621 Villeurbanne cedex, FRANCE
{ludwig.seitz, jean-marc.pierson, lionel.brunie}@liris.cnrs.fr

Abstract

In this article we present grids as an architecture for medical image processing and health-care networks. We argue that confidential patient data should not be stored on the grid unprotected and explain why access control systems do not offer sufficient protection alone. Effective protection can be achieved by storing confidential data in encrypted form. Our proposal details a key management architecture, that allows encrypted storage and still enables users to access decryption keys for data they are authorized to see. Furthermore our architecture is robust against breakdowns, and denial of service attacks. It scales well with the number of users and does not introduce a single point of failure into the system.

Keywords: encrypted storage, key management, medical grids, health-care networks

1 Introduction

Modern medical imagery requires an increasing amount of computing power and data storage capacity. Grids are a promising architecture for providing medical researchers with both resources. Furthermore they could also be used to provide access to medical records in health-care scenarios. However grids often use open networks and untrusted resources to perform their services. Therefore storage of confidential medical data on a grid is problematic. Unauthorized persons must be prevented from gaining access to this data. Access control methods as presented in [1] and [2] achieve this task, as long as an attacker does not gain system administrator access. Therefore it is necessary to take further measures to secure medical data on a grid. Storing the data in encrypted form considerably reduces the risk of disclosure. However since the authorized persons must still be able to access the data, mechanisms must be provided to grant them access to the decryption keys. We propose an architecture that allows storage and access to encrypted data in this environment. Access to decryption keys is granted based on the grids data access permissions.

[*]This work is supported by the Région Rhône-Alpes project RAGTIME (`http://dionysos.univ-lyon2.fr/ miguet/ragtime/`) and the French ministry for research ACI-GRID project (`http://www-sop.inria.fr/aci/grid/public/`)

The rest of this article is organized as follows: In section 2 we present our environment and the objectives our architecture has to achieve. Section 3 presents our proposal for a key management architecture. We discuss the drawbacks and advantages of our system in section 4 and conclude in section 5.

2 Objectives

Grid architectures allow users from different organizations to share heterogeneous resources like computing power, storage capacity and data. The advantage of having such a combined resource pool is obvious, as no single organization could afford the provided resources alone. The idea to use grids for medical applications is becoming widespread [3], [4], [5]. Grids could be used not only for medical image treatment, but also as a health-care network, where they could provide doctors with the medical records they need to treat a specific patient, even if this data is stored at a distant organization.

However storage of confidential data on a grid is at best problematic. As a grid architecture hides the complexity of resource management from the user, he has no way to know on which server his data will be stored. We must therefore assume, that the storage servers are untrusted. Even if we would trust the server to run an un-corrupted access control service to protect our data, hackers might still be able to gain either physical or administrator access to the storage device. In this case they can deactivate the access control mechanisms and still gain access to our data.

Medical data grids will never be created if the security of the data that is to be stored on them can not be guaranteed. Therefore additional protective measures must be taken when dealing with medical data. Bulk data encryption offers a solution to this problem. Even an administrator would not be able to read our encrypted files, unless he had access to the decryption keys. However a new problem arises. The authorized user must have access to the decryption keys. In a health-care scenario, we do not know at the time the data is created, who will be the authorized users. Imagine permissions like this one: *I authorize the* medical personnel *of surgical service of the Lyon central clinic to access all* surgical records *of patient John Doe.* In such cases even when the permission is issued one could not be able to determine the individual subjects (medical personnel) or the individual objects (surgical records) of the permission. Therefore a service is needed that will grant access to the decryption keys to authorized users. This service should not add a new security risk, and therefore no single service should have access to a complete decryption key. Our global goal is therefore to create a scheme that allows the storage of, and access to encrypted data on grid storage devices.

- We want our proposal to be interoperable with most access control mechanisms without functional changes. That means our key management service should not require a change of the grid's access control model. It should be usable as an add-on and should be able to run in mixed settings where a part of the data is not stored encrypted.

- For reasons of vulnerability and to spare bandwidth we want to minimize the use of third parties. In cases where a third party is used, we want to minimize the impact in the event it becomes corrupted.

- We want to avoid a centralized administration of decryption keys. Centralized services do not scale well in distributed and grid environments, often becoming bottlenecks and single points of failure. There must be redundant possibilities of accessing the encryption keys to reduce the risk of a denial of access and to provide for a good scalability.

- The user should not have to trust a single key administration service. Since the entity providing this service will probably be outside of the users organization, he has no possibility to control if the service runs correctly and is not corrupted. Therefore a single key administration service should only have access to key shares and not to complete keys.

3 Methods

The key distribution and storage is governed mainly by the knowledge of users and files in access control permissions. If individual users and files are known at the moment a permission is issued, the decryption keys can be transfered to the users securely or stored encrypted with their public key. Only the authorized users would be able to decrypt these messages. The fact of encrypting a key with another one can be called *creating a key lockbox*. This key lock-boxes can be stored on any publicly accessible device, i.e. together with the data on the untrusted servers, since only the authorized users can open them.

If however the users or files are not individually known (i.e. only identified by a group of users or a set of files) at the time the permission is issued, the decryption keys need to be stored with services where authorized users can retrieve them. These services will be called *key servers*. For any medical data that is to be stored encrypted the decryption keys need to be stored on these key servers. This allows us to access them later, if an access permission is granted on the encrypted data. The following example illustrates this requirement: We choose to store the files of patient John Doe's hip operation in encrypted form and we do not store the decryption keys on the key servers. Later a permission grants access to John Doe's surgical files. The files files concerning the hip operation are part of this surgical files, but since the decryption keys are not available the users holding the permission will not be able to access them. To avoid such situations it is required to store the decryption keys of any encrypted medical file on the key servers.

3.1 General access procedure

If a user wants to access an encrypted file, he will first retrieve the encrypted file from the storage servers. He will then check if he already has the decryption keys stored on his system. If that is not the case the meta-data stored with the encrypted files will indicate him, which key servers he can contact to gain access to the key.

The key servers shall be able to determine a users access permission, to decide if he is to be granted access to some decryption key. Therefore the key servers must use the grid access control service for example to verify a users group membership or if a file is part of a certain set. Once a users permission has been determined the key servers grant him access to the decryption keys and he can then use them to access the file. Figure 1 illustrates this access procedure.

3.2 Revoking permissions

When an access permission is revoked, the user may be able to keep copies of the unencrypted files or even the keys he had access to. Even if this is not the case, there is no way to prevent him from disclosing the content of the files once he has accessed it.

However it may be required to prevent users that no longer have access permissions from changing the files or reading new versions of it. There are several ways to do this. The first is to re-encrypt a file, when an access permission is revoked and to update the key on the key servers. This is cumbersome and consumes a lot of computing power.

If the storage sites have an access control mechanism, we can simply deny the access to the encrypted file on the storage site and leave the encryption key unchanged. The former user may still get access to the data, if he gets administrator access to the storage device.

An intermediate way would be to do a lazy update, where re-encryption is delayed after a permission change, until the concerned file is actually changed.

We believe that there is no best solution to the revocation problem. Therefore the system should provide owners of encrypted files to choose with the option of the three presented solutions he wants to use. As our approach is designed to be used in concert with a classical access control mechanism, the third option is to be used to minimize consumption of computing power while still maximizing security.

3.3 Avoiding single points of failure

In a grid with lots of requests a single key server may become a communication bottleneck and even a single point of failure. If a single key server holds the keys to all objects in our environment, it will be a prime target for attackers willing to compromise the system or more simply to run a denial-of-service attack. It is therefore reasonable to operate a network of distributed key servers.

Our proposal is to use a network of independent key servers and to have them store shares of our decryption keys. Classical secret sharing mechanisms like [6] can be used to divide single keys between several key servers. An arbitrary number m is chosen at the time a key is to be stored. Then a number of key shares $n \geq m$ is produced from that key. These shares are distributed to n different key servers. If an authorized subject wants to recover the key, he has to gather m different key shares from the key servers. There is no risk of key disclosure, as long as less than m key servers are compromised. Figure 1 illustrates data access with the use of key shares. Note that *any* combination of m different key shares will enable a subject to reconstruct the key. Thus an object administrator wishing to guaranty access to the key even if b key servers break down, simply has to chose the number n of key shares he generates as $n = m + b$ and distribute them on n different key servers.

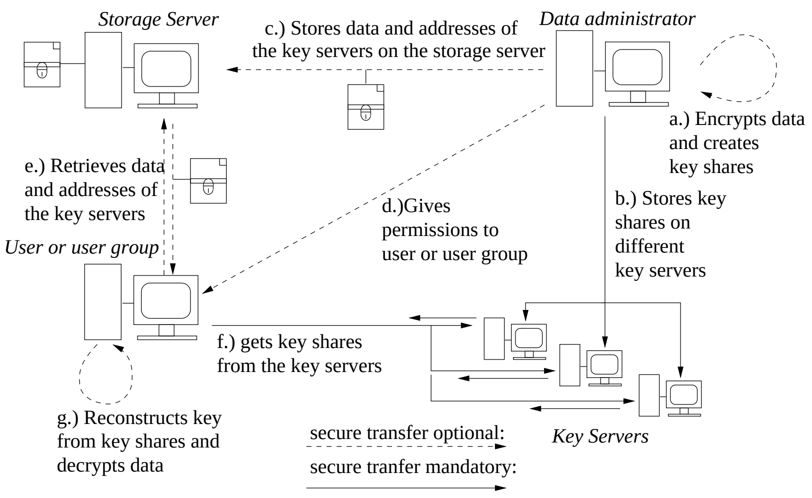
Storage Server
c.) Stores data and addresses of the key servers on the storage server
Data administrator
a.) Encrypts data and creates key shares
e.) Retrieves data and addresses of the key servers
User or user group
d.)Gives permissions to user or user group
b.) Stores key shares on different key servers
f.) gets key shares from the key servers
g.) Reconstructs key from key shares and decrypts data
secure transfer optional:
secure tranfer mandatory:
Key Servers

Figure 1: Access procedure when shares of the decryption keys are stored on different key servers.

4 Discussion

In this section we discuss the drawbacks and advantages of the proposed system, with special attention to the grid computing environment.

The drawbacks of the system come principally from the functions that have to be performed by the key servers.

- Contacting different key servers before being able to decrypt a file slows down the access procedure.
- As the key servers are partially trusted third parties, an attacker may gain access to encryption keys if he manages to corrupt multiple key servers. Setting the number of key shares needed for reconstruction high enough and careful administration of the key servers should minimize this risk.
- For providing links to the key shares some amount of meta-data needs to be stored together with encrypted data. However most infrastructures for data storage in grids provide ways to store meta-data anyway. Therefore this should only be a minor administrative problem.
- The key servers have to manage access to the key shares and thus control the access to the encrypted files. As the access decisions need to be coherent with the grid's access control system it may be necessary to update the key servers access control information, every time an access permission is issued or changed. However all current access control mechanisms adapted to distributed or grid environments [1], [2], [7], [8] allow decentralized access permission checking. We therefore believe that this problem will only appear with ill-chosen access control mechanisms.

The system has the obvious advantage of eliminating the need to trust the storage servers. Additionally we have found the following advantages to our design:

- Aside the transfer of the meta-data to locate corresponding key servers, the design requires no additional interaction with the storage servers for decryption key access.
- The size of the meta-data required on the storage servers is minimal (addresses of the key servers).
- The system is designed to be interoperable with common access control mechanisms. It can be used optionally and does not exclude unencrypted storage of less sensitive files on the same system.
- Together with grid access control mechanisms the key-server architecture allows for distributed and fine grained access control decisions.
- The key sharing mechanism makes the system tolerant to limited key server breakdowns. It also ensures that the corruption of a single key server will not create a major security breach.

We think that the advantages provided by our architecture outweigh the loss of performance it will cause. However one must keep in mind that it is always difficult to evaluate performance loss against additional security gained. This is also the main reason why our design enables data owners to choose which tradeoff is the best for their needs.

However as most end users will be medical staff untrained with the specifics of computer security, an easy to use interface with good default security settings shall be provided to enable the easy and secure usage of our system. This may also be a plug-in to a standard grid interface designed for medical applications. To our current knowledge yet no such interface exists.

5 Conclusion and future work

We have presented the problem of managing keys for encrypted data storage in grid environments. We detailed the problems arising from user groups and data sets administrated by multiple authorities from different organizations.

Our key management architecture is adapted to those problems and was designed to be easy to add onto common grid access control systems. It can manage variable granularities of access permissions and can be parameterized to fit the security needs of different data administrators.

The MEDIGRID[1] project is a grid project that aims to explore the use of the grid technologies for tackling the processing of huge medical image databases available in the hospitals today. We plan to implement the presented key management architecture for the MEDIGRID project, using the hDSEM[9] and the grid access control system proposed in [1]. With this implementation we will be able to measure the delay caused by contacting the key servers. Another interesting question would be to determine reasonable parameters for the settings of our system (number of key shares, number of key servers, number of data replicas etc).

[1]http://creatis-www.insa-lyon.fr/MEDIGRID

References

[1] L. Seitz, J. Pierson, and L. Brunie. Semantic access control for medical applications in grid environments. In *Euro-Par 2003 Parallel Processing*, volume LNCS 2790, pages 374–383. Springer, 2003.

[2] R Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, Á. Frohner, A. Gianoli, K Lörentey, and F. Spataro. VOMS, an authorization system for virtual organizations. In *Proceedings of the 1st European Across Grids Conference*, 2003.

[3] J. Declerck. Large-scale distributed mammogram analysis: Mammogrid and ediamond. In *Proceedings of the Healthgrid conference*, 2003.

[4] G. Lonsdale. Medical simulation services via the grid. In *Proceedings of the Healthgrid conference*, 2003.

[5] P. Sloot. A distributed medical support system for drug therapy of hiv infection. In *Proceedings of the Healthgrid conference*, 2003.

[6] A. Shamir. How to share a secret. In *Communications of the ACM*, volume 22, pages 612–613, 1979.

[7] K. Keahey and V. Welch. Fine-grain authorzation for resource management in the grid environment. In *Proceedings of the 3rd International Workshop on Grid Computing*, 2002.

[8] D. Chadwick and A. Otenko. The permis x.509 role based privilege management infrastructure. In *Proceedings of the seventh ACM symposium on Access control models and technologies*, 2002.

[9] H. Duque, J. Montagnat, J. Pierson, L. Seitz, L. Brunie, and I. Magin. An architecture for large scale and high performance medical imaging applications. Available from http://hectorduque.free.fr/recherche/tdPapers.html.