A Distributed Kerberized Access Architecture for Real Time Grids[1]

A. Moralis, A. Lenis, M. Grammatikou, S. Papavassiliou, B. Maglaris

Network Management & Optimal Design Lab (NETMODE)
National Technical University of Athens (NTUA)
9 Iroon Polytechniou str. Zografou, 15780, Athens, Greece
{amoral, anglen, mary, papavass, maglaris}@netmode.ntua.gr

Abstract. Authentication, authorization and encryption in large scale distributed Grids are usually based on a Public Key Infrastructure (PKI) with asymmetric encryption and X.509 – Proxy certificates for user single sign-on to resources. This approach, however, introduces processing overhead, that may be undesirable in near real time Grid applications (e.g. Grids used for time critical instrument monitoring and control). To alleviate this we introduce in this paper a Symmetric Key – Kerberos based approach that scales in large Grid environments. We present a Use Case Scenario to test and validate the proposed Architecture, in case of numerous time-critical requests running in parallel.

Keywords: Kerberos, Distributed Systems, Grids, Security, Certificates, AAI, Real Time Control

1. Introduction

Our distributed Kerberized Access Architecture (DKAA) aims at extending Grid middleware to provide a high performance (enabling real time or near real time interactions), but yet secure Authentication and Authorization Infrastructure (AAI) for distributed systems. A remote, distributed over the net, interactive and multi-user system is subject to threats encountered in resource-sharing systems. These include unauthenticated and unauthorized access to resources, session hijacking, replay attacks, Denial of Service (DoS) attacks, etc.

DKAAs' main goal is to provide a balance between a mechanism that offers access control, confidentiality and non-repudiation by authentication, authorization and encryption - thereby eliminating most of the threats mentioned above - for low latency usage of large distributed Grid resources. Therefore, the security architecture should specify a light-weight scalable Authentication & Authorization Infrastructure (AAI), along with efficient procedures for secure message exchanges.

1 This work was supported by the EC GridCC Project (IST 511382). The authors performed this work within IASA (Institute of Acceleration Systems & Applications), a joint Institute of the University of Athens & NTUA

DKAA was designed as the AAI and secure data exchange of the GridCC project [4]. GridCC aims at developing a Grid framework that enables the remote control of instruments with real time or near real time constraints. Their successful operation often requires rapid interaction with computing – storage resources and a large number of instruments. The real time and interactive nature of instrument monitoring and control requires the provisioning of acceptable quality of service among various Grid components. Furthermore, since the shared instruments may belong to different organizations, the implementation of a high performance security scheme that allows dynamic management of Virtual Organizations is of paramount importance.

The paper is organized as follows. Section 2 provides a brief overview of existing popular AAI architectures that can be applied in a collaborative Grid environment, while section 3 describes the proposed Distributed Kerberized Access Architecture. Section 4 defines the basic roles of the various users and the appropriate interactions among the service providers, users and Grid resources. In section 5 the Use Case of DKAA in GridCC is presented, and finally Section 6 concludes the paper.

2. AAI Infrastructures for Grids

Grid Security Infrastructure [7] (GSI) is the de-facto Authentication protocol used in Grid environments [8]. It is based on Public Key Cryptography [9] (PKI), using X.509 certificates. An important extension to the PKI is the use of proxy certificates. Through proxy certificates, single sign-on and delegation is achieved. The authorization is not part of GSI and it is performed independently at the level of Computing Elements (cluster of Grid resources, e.g. instrument controller in GridCC) using CAS [13]. The Trusted Third Party is the Certification Authority (CA) that signs the user and hosts certificates [1]. Secure message exchange can be achieved through encryption. GSI is easily deployed and managed, as each organization manages its users' certificates. Therefore trust among the different organizations is only required. Alternatively, authorization in Grids can be achieved by the employment of Virtual Organization Membership Service (VOMS) [11]. VOMS provides to users an attribute certificate that exposes the roles and the capabilities of each user within his/her VO [12].

A drawback of GSI and VOMS, as in every Public Key based infrastructure, is the cost of maintaining revocation lists of users that are no longer trusted. Furthermore, another drawback of GSI, especially in applications that require low latency, if used exclusively, is the heavy overhead of public key cryptography that it employs. If an entity that has real time requirements accepts numerous requests, the nature of asymmetric (public) key cryptography can prohibit the real time functionality in such environments. This is one of the main observations that have motivated the design of DKAA. Nevertheless, GSI, as a PKI based system, is suitable for a Grid environment, with a multitude of users and resources, geared towards batch processing.

3. Description of Distributed Kerberized Access Architecture (DKAA)

DKAA is designed to offer the same functionality that GSI presents, with the extension of handling authorization in the security layer and not in the application layer. In Figure 1, the general DKAA architecture, along with its basic elements and components are presented.

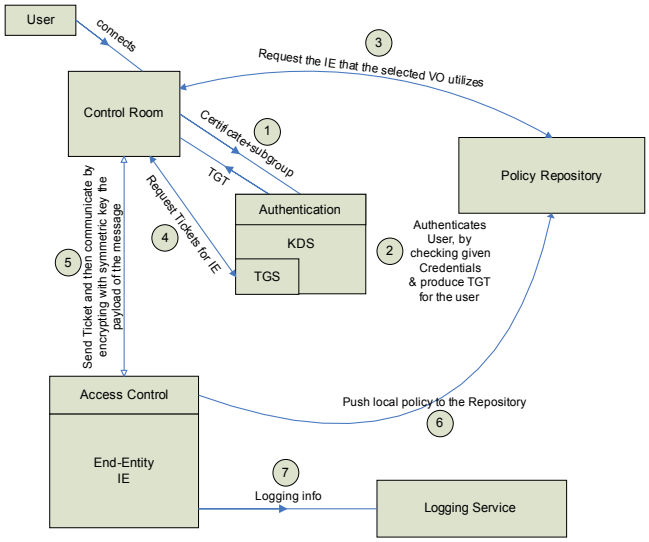

Figure 1. Architecture Overview

The security architectural choices stem from the need to cover distributed environments that give emphasis on providing secure real time (or near real time) access to resources (e.g. instruments) for monitoring and control purposes, with acceptable Quality of Service (QoS) guarantees.

Kerberos [2] is based on the Needham-Schroeder Protocol. Kerberos offers authentication and single sign-on features using symmetric key cryptography. The authentication process is performed by the Authentication Server (AS) in conjunction with the Ticket Granting Server (TGS). Both of these servers run as services within the Key Distribution Server (KDS). In addition, there are mechanisms that map X.509 certificates to the Kerberos Authentication System, in order to maintain interoperability with certificate based systems. PKINIT [6] is an Internet Draft that allows the use of certificates for authentication to the Kerberos system. Furthermore, the API to access Kerberos system is defined in detail in [3].

Once authenticated, a Ticket Granting Ticket (TGT) is returned to the user. When a user wants to access a service located at an end-entity, a DKAA aware client requests a ticket from the TGS transparently to the user. The ticket, when presented to the end-

entity performs mutual authentication. Ticket based authentication is based on symmetric key cryptography and thus the authentication setup imposes minimum overhead to the control clusters in contrast to Public Key Cryptography.

In contrast, a PKI asymmetric scheme establishes a secure session after a computationally expensive handshake procedure (SSL - Secure Socket Layer) imposing a penalty factor that varies form 3.5 to 9, as shown in [5]. Although resuming a session via caching of the results of a previous SSL handshake helps reducing the problem, it does not alleviate it completely, as the initial setup introduces performance penalties, especially under heavy load. In some GridCC Use Cases, thousands of instruments might be involved, each requiring secure message exchanges. The problem is more intense if sessions are terminated in a proxy (e.g. Computing Element or Instrument Element) instead of the terminal resources (e.g. "worker nodes" or instruments) whereby a large number of sessions is concentrated in a single point.

Using Kerberos, as a centralized authentication system, renders the authentication procedures relatively easy via the Key Distribution Server (KDS) acting as a Third Trusted Party. Management of user and roles is significantly simpler than a distributed scheme relying on several PKIs with revocation lists difficult to maintain. In order to cope with the fact that KDS may be a single point of failure, failover KDS systems should be used, so that the availability of the distributed system is guaranteed. Similarly, security of the KDS itself is critical, thus it must be well protected and monitored. Based on Kerberos' single sign-on provisioning capability, a user is authenticated once (gives his/her credentials) and uses multiple resources and services over a period of time (e.g. one hour). This function is implemented by the limited lifetime of the Kerberos tickets, issued by a Ticket Granting Server. After expiration of a ticket, re-authentication is needed. Depending on the application, life-times can vary from minutes to even a day. This duration should be determined, based on time statistics of a control session.

An important implementation decision is whether nodes providing Grid services (e.g. Computing Elements, Instrument Elements, Directory Services, Agreement Services) should follow the approach of a "thin" or "thick" server. For performance purposes it is recommended that the security infrastructure, in the cases that enhanced performance is required, should follow the approach of a thin server. By choosing Kerberos as the authentication system, the authentication at the service provider is simplified to just checking for the validity of the ticket (with the use of symmetric cryptography).

Both Kerberos and GSI are used for authentication purposes and session encryption key negotiations. User authorization is tightly coupled with the policy of each site. It is handled by the Service Provider via a separate system, DKAAs' Access Control Manager (ACM). The ACM checks the users' credentials against a local access list created by the site owner. In order to minimize the access rules, users are divided into subgroups and the access rules are referred to subgroups instead of users. All the access rules of all the service providers are uploaded to the Policy Repository. This global view helps to trace problems regarding the authorization in the distributed environment..

DKAA provides a client API that hides the complexity of the Kerberos protocols and manages the secure message exchange, offering different levels of security (requested by the application).

If secure message exchange is required, the client and the service provider can use the session key (incorporated within the ticket) to encrypt the message or part of the message. By default DKAA encrypts only a timestamp. The level of encryption can play a significant role to achieve the performance goal that each system based on DKAA poses. Based on the above criteria and objectives, we outline the guidelines of a baseline security architecture as follows:

1. An end-user is authenticated by using an X.509 Certificate issued by a trusted Certification Authority to the KDS. This provides single sign on functionality. An authenticated user is authenticated once per session and accesses many resources.
2. The core system consisting of clients and service providers (or end-entities) employs a ticket-based authentication mechanism and symmetric cryptography via centralized Key Distribution Servers.
3. Authorization is performed at the service level employing local access rules.

Following this approach, in the remaining of this section, we present and describe the various basic components of the DKAA architecture.

Authentication Server (AS): AS handles the authentication procedure. The client sends a request to AS, along with the user's credentials. These could be a trusted certificate or username – password. Other types of credentials (like one-time passwords) could be used as well. The authentication procedure is encrypted, using SSL. The AS checks the user's credentials and if they are found correct, a TGT is returned. This TGT among others contains user-specific information, referred to as principal. The principal contains the user name and the subgroup that he/she belongs to (called instance in Kerberos terminology). The user credentials are also stored in the AS. Ticket Granting Server (TGS): TGS grants tickets to a requesting service. The client must present the TGT received from the AS. After checking, the TGS grants a ticket for the required service. The ticket contains the user principal. This function is performed transparently to the user. Policy Repository: It collects local policies for all services. The policy of each service is set by the site owner that the service resides. It is used to locate the existence of specific subgroups within the various services. This is especially important for a single sign-on procedure.

Access Control Manager: It is a software layer similar to an application layer firewall. It filters all incoming connections to the service in two stages:

- Authentication of the received message. The first time a user accesses the end-entity, within a control session, he/she sends the ticket he/she owns to access the end-entity. The ACM decrypts it with the end-entity key and stores locally the session key along with the user principal (username/subgroup) and the expiration time of the session key. This is done only once for the duration of the session key. After that, authorization of the messages is performed with the decryption of the content with the session key.
- Authorization to the requested service. The end-entity checks the local access rules set by the site owner, on the requested service; in conjunction with the user's subgroup it accepts or denies access to the service. This is done per request.

Logging Service: It holds the audit logs. Every transaction with the service providers is recorded, along with its status. The status could be "success" or "failure".

A transaction may include, but not limited to, the User/subgroup, the command, the status and the reason.

4. Users, Roles and Access Policy

DKAA is designed to allow users to authenticate to resources that permit real time or near real-time access and control on distributed resources (e.g. instruments). Therefore the overall infrastructure has to support communication and interactions between two types of entities: Users and Resources (e.g. instruments). Each resource provides some functions to the users, which in turn are translated into services. The users control resources by using the corresponding services.

The identity of a user or a service is uniquely identified in the simplest form with the use of a *principal/instance@REALM* (following Kerberos V5 terminology). The use of a REALM permits the existence of the same principals in different domains. A REALM can be viewed as analogous to a domain or a Virtual Organization (VO) in Grid terms. The principal value refers to the username and the instance value refers to the subgroup that the user has log in. It is obvious that each user (principal) can belong to more than one subgroups (instances).

One of the most important steps in deploying the DKAA is to identify the basic roles of the users. Based on these roles we can specify the appropriate interactions among the service providers, users, Authorization Server, Policy Repository and Accounting Server.

Each role is mapped to a subgroup. Two different kinds of subgroups are defined: *Preset subgroups* and *Dynamic subgroups*. Preset subgroups specify pre-defined roles that are primarily used for administration purposes. As a result all users belonging to these subgroups inherit the corresponding privileges and limitations. In the following we provide a short description of the functionality of all the Preset subgroups that will be used within the DKAA.

Administrator: Administrators are users that can add/delete/modify other user profiles. They can also delete a subgroup, even if it does not belong to them. Their job is to maintain consistency between users and corresponding subgroups.

Site Owner: The site owners set the local access policy (authorization) on their sites. The access policy is set on the subgroups and not on the corresponding users.

Subgroup_Coordinator: This group has all the users that can create a dynamic subgroup (dynamic subgroups are defined below). The Coordinators can add users to dynamic subgroup that they have created, add resources to them and finally delete them.

Users: This subgroup includes by default all potential users. In addition, when a user is created, a unique subgroup containing only this user can be created.

As mentioned above, apart from preset subgroups, dynamic subgroups may be required and defined. Each dynamic subgroup defines a specific non permanent task, and contains the appropriate users and services on the corresponding entities. Each dynamic subgroup can be managed by its creator, who is a user that belongs to the Coordinator subgroup.

The creation of a subgroup and the specification of its users and services, do not automatically provide authorization to the resources. The coordinator of the subgroup must explicitly request from the site owners of distributed systems to obtain access to the resources and the services required for the specific task. This is not defined within DKAA.

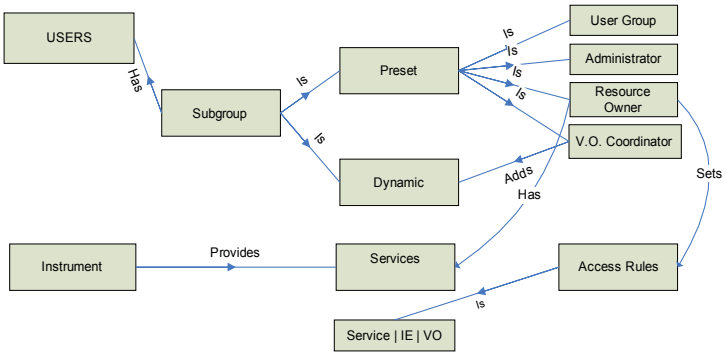

Figure 2. Entities & Roles Diagram

The authorization to invoke a service offered by a specific end-entity must be granted by its site owner. This authorization is given at the subgroup level, and is not provided explicitly to each individual user. This approach is adopted in order to limit the number of access rules that exist within each end-entity and therefore improve the efficiency of the authorization process. However, if granularity down to the user level is required by a use case, a separate subgroup can be created for each user. In this way, the access control can be extended to support access to the user level.

The format of an authorization access rule could be as follows:

Service	Serv Provider ID	Subgroup

The above represents a general form/representation. In case that, services are provided in the form of a Web Service [10], the rule can be refined to fit the Web Service architecture as follows:

Service: portType, Operation

Serv-Provider ID: Service Endpoint URL orService Location

Subgroup: Kerberos Instance or Group

Therefore, according to the above terms an access policy rule may take the following form:

Operation	PortType Service	Endpoint Url	Kerberos Instance

The default access policy is default deny. This means that if specific access rules are not explicitly defined, the service is denied to any subgroup requesting it. Therefore effectiveness and simplicity is maintained in enforcing end-entity access rules. If complicated access policies are needed, these must be first translated to access rules of the above format, which in turn will be enforced to the service provider. However this translation is not to be provided by the DKAA, and should be performed by the authorized site owner, if required.

It should be also noted that a user may belong to more than one subgroup. This may happen, if a user is involved in more than one experiment or has more than one

predefined roles. In this case, each instance of the user must refer to only one active subgroup.

The above defined entities as well as their interactions are presented in figure 2.

5. The GridCC Security Use Case

As mentioned earlier, DKAA is being developed as the Authentication & Authorization Infrastructure (AAI) of the GridCC project [4]. To this end a new resource has been defined, namely the Instrument Element (IE), similar to the Computing and Storage Elements (CEs, SEs) of the batch Grid. The IE acts as the middleware that enables the remote control of instruments over the network as services. With the addition of supporting services, like the Workflow service and the Agreement service, a user of the system will be able to automatically control instruments and define the Quality of Service characteristics of his/her interactions. To offer simplicity to the users, a Virtual Control Room has also been defined that acts as the User Interface of all the services in the GridCC collaborative environment.

In the following we demonstrate the operation of the basic elements of the DKAA architecture in the GridCC security Use Case, by identifying and describing the various interactions and exchanges involved during the authentication and authorization phase. In a typical GridCC scenario there are at least two different organizations that have IEs. Let us consider the scenario illustrated in figure 3, where all other supporting GridCC entities have been omitted for simplicity in the presentation.

In this scenario we assume the existence of two organizations: Organization1 and Organization2. These organizations are independent in every aspect, including policy and users. Organization1 has two Instrument Elements, IE1 and IE2, while Organization2 has only one, IE3. The two organizations have agreed to share part of their resources to users belonging to Virtual Organization 1 (VO1). Specifically, Organization1 provides IE1 and Organization2 provides IE3. VO1 also sets up a Key Distribution Server (KDC) that has all the credentials (GSI certificates) of the users belonging to VO1.

For every user (certificate) a password is created. Also passwords are created for IE1 and IE2. These passwords are only known to their corresponding users or IEs and the KDC. Let us suppose that user@VO1 tries to send control commands to IE1 and IE3. The steps involved in this exchange are:

1. The user logs in the Control Room. The user submits his/her credentials (GSI certificate) and the subgroup that he/she wants to use. Transparent to the user, the client API logs in to the KDC1 using the user's certificate. In the case that the credentials are valid, the log in, is successful and a TGT is returned.
2. Let us assume that the user sends commandA and commandB to IE1 and IE3 respectively, through the Control Room interface, choosing the encryption scheme (whether the whole message or part of it, is encrypted). Then, transparently to the user the following actions take place:

- a. the client API requests tickets for IE1 and IE3 from the TGS by providing the TGT that was acquired in step 1.
- b. the client API communicates with the IE1 and IE3 and sends the ticket.
- c. IE1 and IE3 check the ticket. If it is accepted the user is authenticated and the Session Key (SK) is stored locally at the ACM of IE1 and IE3, (SK1 for IE1 and SK3 for IE3) along with the user and his subgroup. The session key is valid until its lifetime expires.
- d. the client API sends commandA to IE1 using the selected encryption scheme. The key that is used is the SK1. Similarly SK2 is used to send commandB to IE3.
- e. The ACM of IE1 (or IE3) uses the session key SK1 (or SK3) to decrypt the command.
- f. The ACM checks the local rules if the user's subgroup has access to commandA (or commandB) and accepts or denies the corresponding command.
- g. In case of success a command "successful" is returned to the CR, otherwise the command "failure" is returned.

3. The Control Room displays the result of the user's commands (success or failure).

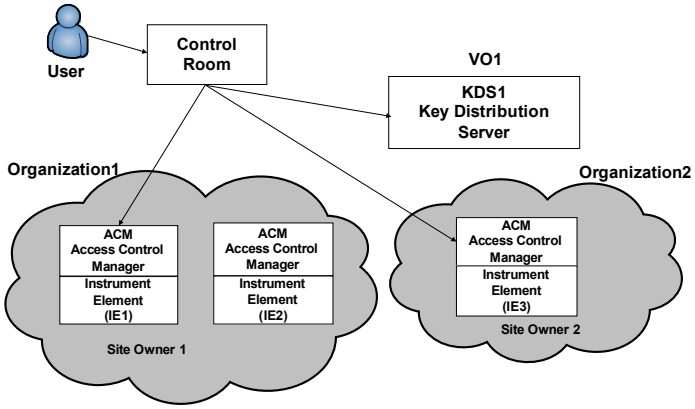

Figure. 3. GridCC Use Case

4. When the user send another command to IE1 or IE3 only steps d to g are performed for the lifetime of the Session Key. In the scenario described above, it is assumed that the access rules have been set locally by the site owner of each Organization. This step is performed prior to the creation of the subgroup. If there are any changes in local policies, they are pushed to the Policy Repository. Access rules that allow access for the subgroup of users are set only in IE1 and IE3. In the case that the user sends a command at IE2, although he/she is authenticated, the lack of an access rule for his subgroup at IE2, results in denying access to it.

6. Conclusions and Future Work

In this paper we introduced a Distributed Security Infrastructure, namely DKAA that aims to meet the real time constraints in interacting with a large number of Grid resources (e.g. distributed monitoring and control of time critical instruments). This proposed infrastructure is based on Symmetric Key Cryptography that has low demands on CPU power, thus allowing the overall service to offer lower response times compared to alternative Public Key Cryptography approaches. It should be noted that in this paper our emphasis was placed on the introduction, definition and description of the basic principles, the guidelines and the framework of the proposed DKAA architecture. However, it is part of our current and future work within the GridCC project, to develop and provide a toolbox that can enable generalized Grid Services to use this infrastructure.

References

1. The European Policy Management Authority for Grid Authentication in e-Science, http://www.eugridpma.org/
2. RFC 1510- The Kerberos Network Authentication Service (V5)
3. RFC 1508 - Generic Security Service Application Program Interface
4. GRIDCC Project web site - www.gridcc.org
5. C. Coarfa, P. Druschel and D.S. Wallach, “Performance Analysis of TLS Web Servers”, 9th Network and Systems Security Symposium, pp. 553--558, 2002
6. PKINIT Draft - http://www.ietf.org/internetdrafts/draft-ietf-krb-wg-ocsp-for-pkinit-06.txt
7. Grid Security Infrastructure - http://wwwunix. globus.org/toolkit/docs/3.2/security.html
8. Global Grid Forum - http://www.ggf.org/
9. RFC 2459 - Internet X.509 Public Key Infrastructure Certificate and CRL Profile.
10. W3C Web Services Activity http://www.w3.org/2002/ws/
11. R. Alfieri et al, “VOMS, an Authorization System for Virtual Organizations”, Presented at the 1st European Across Grids Conf., Santiago de Compostela, Spain, Feb. 14, 2003
12. R. Alfieri, R Cecchini, V. Ciaschini, F. Spataro, L. Dell'Agnello, A. Frohner, K. Lorentey, “From gridmap-file to VOMS: managing authorization in a Grid environment”, Future Generation Computer Systems. Vol. 21, no. 4, pp. 549-558. Apr. 2005
13. L. Pearlman, V. Welch, I. Foster, K. Kesselman and S. Tuecke, “A Community Authorization Service for Group Collaboration”, IEEE Workshop on Policies for Distributed Systems and Networks, 2002