PROCEEDINGS OF SCIENCE

A new “lightweight” Crypto Library for supporting a new Advanced Grid Authentication Process with Smart Cards

Giuseppe La Rocca[1]
Italian National Institute of Nuclear Physics, Division of Catania
Via S. Sofia 64, 95123 Catania, Italy
E-mail: giuseppe.larocca@ct.infn.it

Roberto Barbera
Department of Physics and Astronomy of the University of Catania and INFN
Viale A. Doria 6, 95125 Catania, Italy
E-mail: roberto.barbera@ct.infn.it

Vincenzo Ciaschini
Italian National Institute of Nuclear Physics, CNAF
Viale Berti Pichat 6/2, 40127 Bologna, Italy
E-mail: vincenzo.ciaschini@cnaf.infn.it

Alberto Falzone
NICE srl
Via Milliavacca 9, 14100 Asti, Italy
E-mail: alberto.falzone@nice-software.com

Salvatore Monforte
Italian National Institute of Nuclear Physics, Division of Catania
Via S. Sofia 64, 95123 Catania, Italy
E-mail: salvatore.monforte@ct.infn.it

In this paper we present how X.509 compliant “robot” certificates stored on smart cards can be used to enhance the security architecture of the gLite Grid middleware .The solution we propose extends the native Java ™ Cryptographic Token Standard Interface (PKCS#11) libraries with the Bouncy Castle and the Java CoG Kits API in order to implement a “lightweight” crypto utility which may be used by single users, applications, general purpose web portals and/or Science Gateways to create valid proxies by accessing the digital certificates stored on a smart card.

The International Symposium on Grids and Clouds and the Open Grid Forum
Academia Sinica, Taipei, Taiwan
March 19 - 25, 2011

[1] Speaker

PoS(ISGC 2011 & OGF 31)029

1. Introduction

The intensive research, co-funded by the European Union within the fifth, sixth and seventh Framework Programme for Research and Technological Development, have seen, after almost one decade, the creation of the European e-Infrastructure which is now mature enough to offer around the clock a production quality service to a large number of Virtual Research Communities (VRCs) gathered in more than 200 Virtual Organisations (VOs). The production infrastructure is now being used daily hundreds of applications. On average, over 28 million jobs are being served every month. The development of these Grid sites, both in size and number, has also been accompanied by the deployment of different Grid middleware which provide users with high level services for scheduling ad running computational jobs, accessing and moving data, and obtaining information on the Grid infrastructure as well as Grid applications. In this paper we focus on the security framework of the gLite [1] middleware, a middleware stack developed in the context of the EGEE project [2] that combines components developed by various related initiatives projects, namely Condor [3], Globus [4], W-LCG [5], and VDT [6].

The existing Grid middleware, and in particular gLite, rely on the adoption of a Public Key Infrastructure (PKI) [7,8] consisting of X.509 compliant digital certificates for user authentication and Certification Authorities (CAs) mutually trusted by international Policy Management Authorities (PMAs). These credentials must be present on each User Interface (UI), i.e. the client used by the user to access the computational and storage resources available on the Grid. The exposure of the certificate's private key on multiple locations is considered a security weakness, as the certificate may be subjected to possible fraudulent use by non-authorized people logged to the same UI (e.g., the system administrator). Public-key cryptography implemements the concept of a digital signature providing a practical, elegant mechanism for symmetric key agreement. Through a PKI, it is possible to verify and validate the identification of a given user requesting access to a given computational and storage resource and enable secure communications over open and untrusted networks such as the Internet. The basis characteristic of a public key system is that each entity possesses two keys: a public and a private one. The public key is published and known by everyone while the private key is only known by the owner and kept secret in a safe place. Public keys are used to encrypt messages and verify signatures, while private keys are used to decrypt messages and make signatures. If someone else knows the entity's private key, he/she can not only read confidential messages sent to the owner, but also impersonate himself/herself in electronic communications such as email, electronic transactions, etc. In order to avoid the fraudulent use of private keys, users must therefore protect their private keys adequately.

Moreover, there is a total lack of support for other authentication mechanisms such as smart cards even if this hardware, with its properties, can help in keeping safe these certificates and avoid any fraudulent use. They are usually tamper-resistant devices that can be easily connected to a laptop and used to store private keys and enhance the protection of private credentials. To access private objects stored into the smart card, a pin-code is requested. An additional protection is given to private keys and secret keys which are marked as *"sensitive"* or

"un-extractable". Sensitive keys cannot be revealed in plain text off the token and un-extractable keys can not be revealed off the token when encrypted.

In this paper we describe the work done in the last yar to design and implement a new Grid authentication method based on the use of digital certificates stored on smart cards. The solution we propose extends the native Java ™ Cryptographic Token Interface Standard [9] (referred to as PKCS#11) with the Java CoG Kits (ver. 1.8.0) [10] and the Bouncy Castle [11] API in order to implement a "lightweight" crypto utility which may be used by by single users, applications, general purpose web portals and/or Science Gateways to create valid proxies by accessing the digital certificates stored on a smart card. Another approach, based on the use of a wrapper script (mkproxy) to read the credentials stored on a smart card and create a Grid proxy is documented in the following reference [12]. From a technical point of view, the solution developed at NIKHEF [13] combines openssl and *pkcs11-tools* commands to create a plain proxy on request. This solution was successfully used in the past to support the bioinformatics group at CNR-ITB [14] Bari, involved in the LIBI project [15], to perform large scale phylogenetic analysis on the distributed computational and storage resources of a Grid infrastructure using the GENIUS Grid portal [16] powered by EnginFrame[17], as shown in Fig. 1.

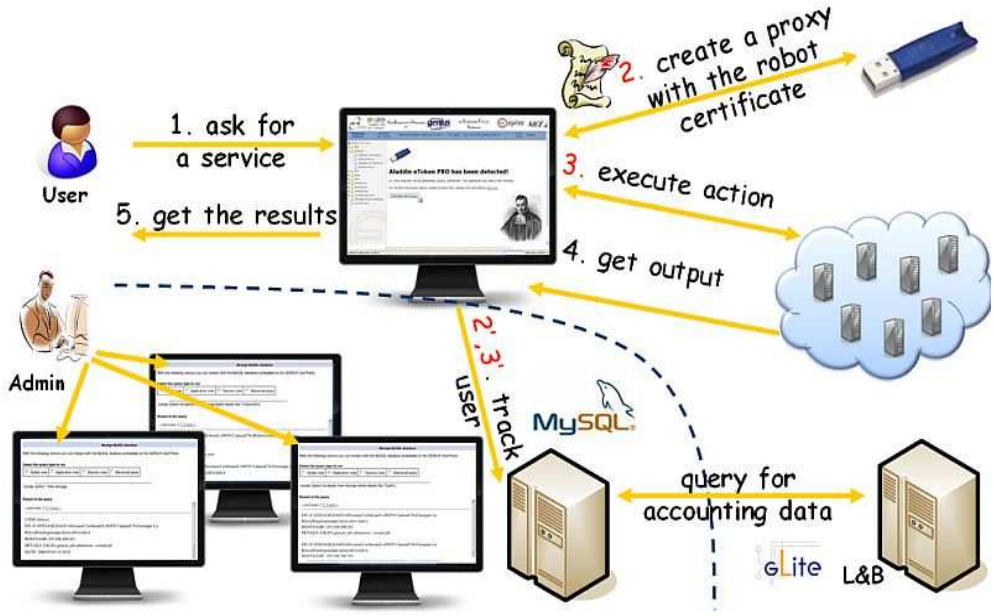

Fig. 1 – Accessing the Grid infrastructure with the GENIUS portal and certificates on smart cards.

In this scenario, the USB token with the digital certificate was plugged into the grid User Interface where the GENIUS portal was running on. Every time users login into the portal, an automatic service drives them through the creation of a temporary proxy certificate. The proxy is generated reading the credentials stored on the robot certificate installed in the USB key. Once the proxy is successfully generated by the portal, users are automatically redirected to the home page of the application associated with the robot's function. The behaviour of all the users who access the Grid with the credentials of the robot certificate is monitored by the User Tracking System (UTS). This work has been published in [18, 19, 20, 21]. Even if this solution

reduced the scientific gap of this scientific community to access the Grid infrastructure and run their applications, from a technical point of view presents a couple of limitations for Grid portals and Science Gateways developers. The main limitation of this approach is that this wrapper generates only plain proxies. Before the Grid proxy can be used, the Virtual Organisation Memebership Service (VOMS) has to be contacted and the VOMS Attribute Certificate (AC) attributes have to be appended to the original one otherwise the authorization process on the remote sites will fail. In addition, this solution requires to hack this wrapper in order to be used by last generation of Grid portals and/or Science Gateways.

The new "lightweight" crypto library we present in this paper overcomes these limitations as the proxy generation process, with the right VOMS AC attributes, is performed directly via a programmable interface. The core of the crypto library is developed using Java APIs and, for this reason, it can be better used with any web-based Grid portals and Science Gateways. In section 2 we provide an overview of PKCS#11 which is a standard for "cryptographic tokens". The overview is intentionally high level and brief, touching only on aspects directly relevant to an understanding of the native programming interface. For a broader and more thorough discussion, the reader is invite to have a look at the official reference guide [9]. In section 2 we will also introduce the Java ™ PKCS#11 implementation, the Bouncy Castle APIs and the Java CoG Kits, while in section 3 the details of our implementation will be described.

2. Overview of the Java ™ PKCS#11 Implementation

The Java™ PKCS#11 implementation defines native programming interfaces to cryptographic tokens such as hardware cryptographic accelerators and smart cards. It defines sixty prototypes for functions (referred to as *cryptoki* library) that together can be used to perform a wide range of cryptographic mechanisms, including digital signatures, public key ciphers, symmetric key cipher, hash functions, etc. In order to make the integration of native PKCS#11 tokens into the Java platform easier, a new cryptographic provider, the Sun PKCS#11 provider, has been introduced into the J2SE 5.0 release. This new provider works as a bridge between the Java Cryptography Architecture (JCA), the Java Cryptographic Extension (JCE) and the native PKCS#11 cryptographic API, translating the calls and conventions between the two. No modifications to the application are requested. The only requirement is the proper configuration of the provider into the Java Runtime Environment.

2.1 Requirements

The Sun PKCS#11 provider is supported on Solaris SPARC platforms (both 32 and 64-bit Java VMs) and on x86 compatible platforms (Solaris, Linux and Windows operating systems). It is not supported, however, on 64-bit AMD64 and Itanium platforms.

2.2 Configuration

The Sun PKCS#11 provider is implemented by the main class sun.security.pkcs11.SunPKCS11 and accepts the full pathname of a configuration file as an argument. The installation of the provider can be done either statically or programmatically. To install the provider statically, the user has to add the provider to the Java Security properties file

($JAVA_HOME/lib/security/java.security). As an example, here follows a fragment of the java.security file that installs the SUN PKCS#11 provider with the configuration file /opt/bar/cfg/eToken.cfg.

```
# Configuration for security providers 1-6 omitted
Security.provider.7=sun.security.pkcs11.SunPKCS11 /opt/bar/cfg/eToken.cfg
```

The configuration file is a text file that contains entries in the attribute=value format:

```
$ cat eToken.cfg
name = eToken
library = /usr/lib/libeTPkcs11.so
```

The two mandatory attributes are name and library. For a complete list of values and attributes, please refer to the official reference guide [9].

To install the provider dynamically, one has to create an instance of the provider with the appropriate configuration filename and install it. Here is a code snippet to configure the provider:

```
// Create an instance of the provider
Provider eToken_PKCS11Provider = new sun.security.pkcs11.SunPKCS11 (TOKEN_CONFIG_FILE);
Security.insertProviderAt (eToken_PKCS11Provider,1);
try {
   keyStore = java.security.KeyStore.getInstance ("PKCS11", eToken_PKCS11Provider);
} catch (java.security.KeyStoreException KeyStoreException)
{ log.error (" KeyStoreException= " + KeyStoreException); }
```

2.3 Bouncy Castle

Bouncy Castle is a collection of Java and C# APIs for cryptography. These lightweight APIs work almost with everything, from J2ME to JDK 1.6. In our implementation, these APIs have been used to create version 3 of X.509 certificates with the certificate's attributes we collect from the etoken.

2.4 The Java CoG Kits

The Java CoG Kits, distributed under the Globus Toolkit Public License (GTPL), is an extension of the Java libraries and classes which can be used to provide Globus Toolkit functionalities within their code without calling scripts,or, in some cases, without having Globus installed. In the present work these APIs have been used for interfacing with the MyProxy server and provide the possibility to store long-term proxy delegators.

3. The Implementation

In the current implementation, the library runs with the Aladdin [22] eToken PRO 32 Kbytes directly plugged to a remote server based on Scientific Linux release 4.5 (Beryllium). More information on the implementation is given on the next sub-sections.

3.1 The "lightweight" crypto library overview

The new "lightweight" crypto library has been designed and developed considering the native Java™ PKCS#11 cryptographic API (ver. 2.0), the Bouncy Castle and the Java CoG Kits APIs (ver. 1.8.0) and it is distributed under Apache 2.0 license [23]. In the remote server, where the eToken was plugged, the Aladdin's eToken PKI Client software (pkiclient-full-4.55-34) was also installed. This software enables eToken USB operations and the implementation of eToken PKI-based solutions. For what concerns the credentials stored on the smart card, we used robot certificates issued by the INFN CA [24] even if personal X.509 certificates may also be supported. Robot certificates have recently been introduced to address the requirements of those users who want to exploit the huge computational resources but do not want to be bothered with the internals of the Grid Security Infrastructure. The Aladdin eToken is a powerful and secure hardware device that enhances the security of data on public and private networks. It enables the generation and the secure storage of passwords and digital certificates, strong authentication, digital signing and encryption, and more. eTokens can be used to hold secret information, certificates and private keys for use in authentication solutions for LANs, WANs, VPNs, e-commerce and mobile computing. A single eToken can concurrently store a number of private keys, certificates and passwords, for use in a wide variety of applications. It can also generate keys and perform sensitive encryption operations on-chip, ensuring that users' keys are never exposed to the PC environment. The design of the lightweight crypto library was conceived to support multi-threaded client/server connections with SSL/TLS protection. The digital certificate was stored into a smart card which was made available on the server around the clock. The server was configured to support requests for listing the content of the smart card and generate VOMS proxies. In addition, a long-term proxy delegator is automatically registered in a MyProxy server every time a proxy certificate is generated. Figure 2 shows the high-level overview of the "lightweight" crypto library, as we implemented it.

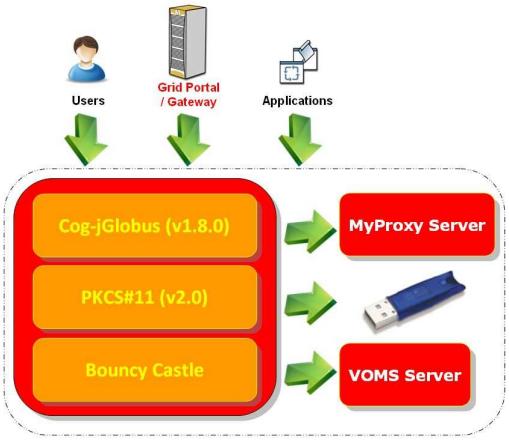

Fig. 2 – Overview of the "lightweight" crypto library.

Since private keys are never exposed outside the eToken, in order to generate a proxy certificate the org.bouncycastle.x509.X509V3CertificateGenerator Java class has been used. The structure describing the proxy nature is composed by a set of nested objects matching ASN.1 definitions. In this work, the proxy structure has been filled up considering the certificate's attributes such as: Distinguished Name (DN), issuer, serial number and lifetime of the proxy (just to name a few) we retrieved from the smart card. This information has been collected using the PKCS#11 native interface. The proxy certificate created on request is,signed with the private key of the digital certificate stored on the smart card. The eToken usually does not allow to expose the private keys outside but allows to perform some operations such as the signature of documents. The plain proxy certificate created at this stage is sent to the calling client and stored in the $PROXYFILE environment variable. Subsequently, a GSI connection with the VOMS server has been set up to append the needed VOMS AC to the original proxy. Last but not least, using the Java CoG Kit, a long-term proxy is registered in the MyProxy server enabling the proxy renewal process. The introduction of this new crypto library improves the scenario as shown in Fig. 3.

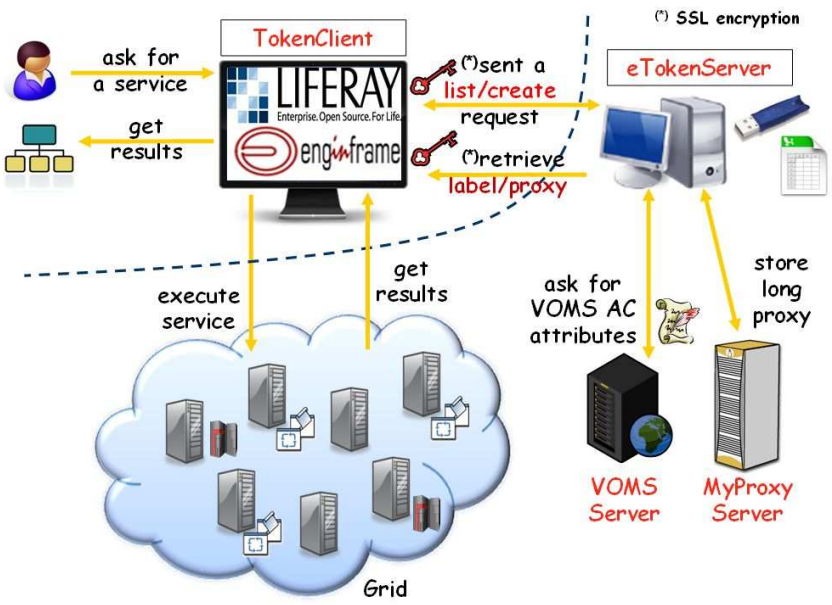

Fig. 3 – The new scenario enabled by the new "lightweight" crypto library.

The core of the new library is represented by the eTokenServer Java class, a multi-threaded server which accepts all the requests coming from a list of authorized clients and manages a list of digital credentials kept inside the USB token. Client requests are implemented by the TokenClient Java class. With this class, single users, applications, general purpose Grid portals and/or Science Gateways can send requests to the eTokenServer either to browse the available X.509 certificates or to generate VOMS proxies. To improve the security between clients and server, the SSL protocol was used to authenticate the communications. The X.509 certificate of the server has been converted in DER format and stored into a Java Key Store (JKS). This certificate is used to authenticate the communication between clients and server. In additional

the firewall of the server is instructed to accept requests from a list of authorized IP numbers. When the server starts, a list of VOMS configuration parameters will be loaded in a memory HashMap and the USB token pin-code is provided in input to complete the start-up. With this configuration parameters, the server will accept and satisfy all the list/create requests.

3.2 The "lightweight" crypto library at work

In the next two sub-sections we report how the crypto library browses the USB token and generates a VOMS proxy with a specific label.

3.2.1 Browsing the eTokenServer smart card

When the client application sends to the server a request for listing, the server answers with the label of digital certificates available on the server (see Fig.4).

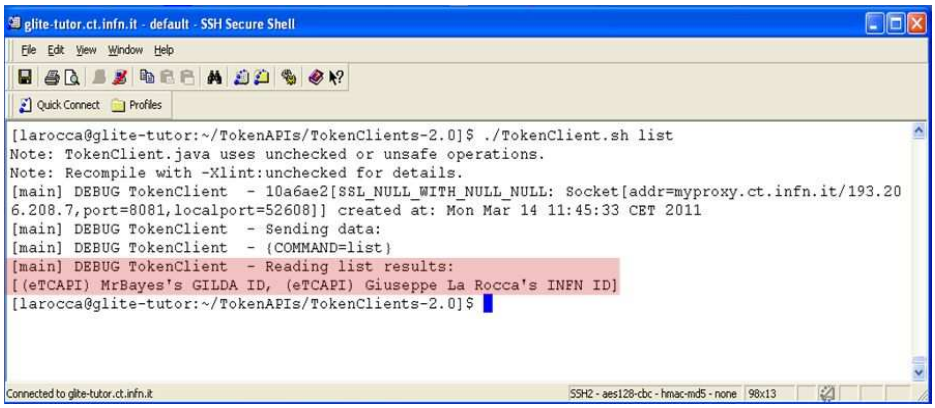

Fig. 4 – The TokenClient sends a "list" request to the eTokenServer.

In this case, two certificates are available on the server.

3.2.2 Create proxy certificate from the eTokenServer

When the client application sends to the server a request to create a proxy, the server reads the USB token, generates the proxy for the given label, uploads the long-term proxy to the MyProxy server and sends back the final VOMS proxy to the caller (see Fig.5).

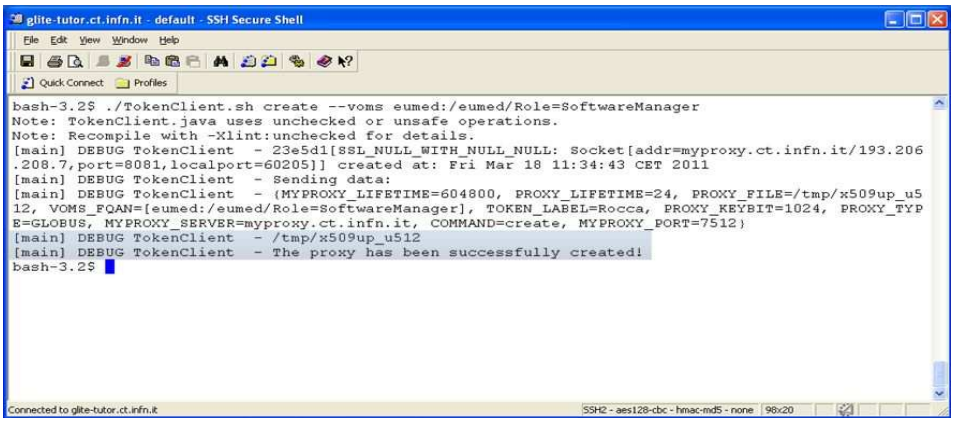

Fig. 5 – The TokenClient sends a "create" request to the eTokenServer.

The first version of this new "lightweight" crypto library has been successfully used by the new e-Collaboration environment based on the high customizable features of Liferay portal and the GENIUS/EnginFrame 2010 Java/XML framework [25, 26].

Conclusions

The Java SE platform provides developers with a rather complete set of security APIs, algorithms, tools and protocols. Among them, we mention the PKCS#11 cryptographic tokens which have been used in this work together with the Bouncy Castle and CoG Kits Java APIs to implement a new security solution for the gLite Grid middleware. The lightweight crypto library we described in this paper can be used by single users, applications, Grid portals and/or Science Gateways to generate VOMS proxies starting from the credentials stored into an eToken-based smart card. The new library will be used in the context of the DECIDE (Diagnostic Enhancement of Confidence by an International Distributed Environment) project [27] to help the medical scientific community to access transparently the Grid infrastructure.

Acknowledgments

This work has been partially supported by the FP7 EGI-InSPIRE project [28]. The authors would like to thank all the people who supported this work contributing with ideas, continuous feedback and cooperation.

References

[1] The gLite middleware – www.glite.org

[2] The EGEE Project web site – http://public.eu-egee.org

[3] J. Frey, T. Tannenbaum, M. Livny, I. Foster, S. Tuecke, *Condor-G: A Computing Management Agent for Multi-Institutional Grids*. Cluster Computing 2002, 5(3):237-246

[4] I.Foster, C. Kesselman and S. Tuecke, *The Anatomy of Grid. The International Journal of High Performance Computing Applications*, 2001, 15(3):200-222

[5] The LCG Project web site – http://cern.ch/lcg

[6] The Virtual Data Grid Toolkit – www.cs.wisc.edu/vdt

[7] C. Adams and S. Loyd, *Understanding PKI: Concepts, Standards, and Deployment Considerations*, Second Edition. 0-672-32391-5, Nov. 2002

[8] R. Rivest R, A. Shamir, and L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the Association for Computing Machinery, 1978

[9] The Java ™ PKCS#11 Reference Guide http://cupi2.uniandes.edu.co/site/images/recursos/javadoc/j2se/1.5.0/docs/guide/security/p11guide.html

[10] The Bouncy Castle web page – www.bouncycastle.org

[11] The CoG Kits web page – http://dev.globus.org/wiki/CoG_jglobus

[12] Using the Aladdin eToken PRO to generate a grid proxies – www.nikhef.nl/grid/gridwiki/index.php/Using_am_Aladdin_eToken_PRO_to_generate_grid_proxies

[13] The Nikhef web site – www.nikhef.nl

[14] CNR-ITB of Bari web site – www.ba.itb.cnr.it

[15] The LIBI Project web site – www.libi.it

[16] The GENIUS Grid portal web site – https://genius.ct.infn.it

[17] The XML/Java EnginFrame framework web site – www.enginframe.com

[18] R. Barbera, G. Donvito, A. Falzone, G. La Rocca, G. P. Maggi, L. Milanesi, *GENIUS Grid Portal and robot certificates to perform phylogenetic analysis on large scale: an experience within the Italian LIBI Project*. Managed Grids and Cloud Systems in the Asia-Pacific Research Community – Lin, Simon C., Yen, Eric (Eds.), 2010, XII, ISBN 978-1-4419-6468-7

[19] R. Barbera, G. Donvito, A. Falzone, G. La Rocca, G. P. Maggi, L. Milanesi, *The GENIUS Grid Portal and Robot Certificates* – Final Workshop of Grid Projects "PON RICERCA 2000-2006, AVVISO 1575", March 2009, Catania – Italy– ISBN: 978-88-95892-02-3

[20] R. Barbera, G. Donvito, A. Falzone, G. La Rocca, G. P. Maggi, S. Vicario, L. Milanesi, *The GENIUS Grid Portal and robot certificates: a new tool for e-Science* — BMC Bioinformatics 2009, 10 (Suppl 6) doi:10.1186/1471-2105-10-S6-S2

[21] R. Barbera, G. Andronico, G. Donvito, A. Falzone, J. J. Keijser, G. La Rocca, L. Milanesi, G. P. Maggi and S. Vicario, *A grid portal with robot certificates for bioinformatics phylogenetic analyses*, Concurrency and Computation Practice & Experience, Special Issues IWPLS 2009 – John Wiley & Sons. Ltd, Nov. 2010, Vol. 23, Issues 3, pages 246-255, doi:10.1002/cpe.1682

[22] The Aladdin web site – www.aladdin.com/etoken/default.aspx

[23] Apache License, Version 2.0 - www.apache.org/licenses/LICENSE-2.0.html

[24] The INFN CA web site – http://security.fi.infn.it/CA/

[25] R. Rotondo, R. Barbera, G. La Rocca, A. Falzone, P. Maggi, N. Venuti, *Conjugating science gateways and grid portals into e-collaboration environments: the Liferay and GENIUS/EnginFrame use case*, in proceedings of the *TeraGrid conference*, 2010, Pittsburgh, Pennsylvania – ISBN:978-1-60558-818-6, http://doi.acm.org/10.1145/1838574.1838575

[26] R. Barbera, G. Andronico and G. La Rocca (Eds.), A Grid Portal as an e-Collaborative environment powered by Liferay and EnginFrame, in proceedings of the *International Workshop on Science Gateways (IWSG2010)*, Catania, Italy, September 20-21, 2010, Consorzio COMETA (2010), ISBN 978-88-95892-03-0 (See also http://documents.ct.infn.it/record/479/files/iwsg10-proceedings.pdf)

[27] The DECIDE project web site - www.eu-decide.eu

[28] The EGI-InSPIRE project web site – www.egi.eu/projects/egi-inspire/