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

Roberto BARBERA(1)(2), Vincenzo CIASCHINI(3), Alberto FALZONE(4), Giuseppe LA ROCCA(1) and Salvatore MONFORTE(1)

(1) INFN – National Institute of Nuclear Physics, Division of Catania, Italy
(2) Department of Physics and Astronomy of the University of Catania, Italy
(3) INFN – National Institute of Nuclear Physics – CNAF, Division of Bologna, Italy
(4) NICE srl – Asti, Italy

ISGC 2011 & OGF 31
19-21 March 2011, Academia Sinica, Taipei, Taiwan
Outline

• Background:
  – the current state-of-the-art of Grid Security;

• Introduction to smart cards and robot certificates:
  • Installation and Configuration;
  • A use case from bioinformatics;

• Introduction to the “lightweight” crypto library
  – Java™ PKCS#11, Bouncy Castle and Java CoG Kits;
  – The Architecture;
  – The list of software packages;
  – Examples;
  – Future works;

• Summary and Conclusions;
• References.
• **Background:**
  – the current state-of-the-art of Grid Security;

• **Introduction to smart cards and robot certificates:**
  – Installation and Configuration;
  – A use case from bioinformatics.
21st Century Research is becoming computationally intensive research.
Background

- Grid technology allows users to share a wide *plethora* of distributed computational resources regardless of their geographical location, but unfortunately...

There are many things to know about Grid services before to start...

Grid security is indeed based on the Public Key Infrastructure (PKI) of X.509 certificates and the procedure to manage these certificates is unfortunately not straightforward;

The adoption of *robot certificates* can reduce these barriers and help non-expert users to experience Grids technology!
Robot certificates in a nutshell

- Robot certificates have been introduced to allow non-users to experience the Grid paradigm for research activity;
  - They are extremely useful, for instance, to automate Grid service monitoring, data processing production, distributed data collection systems;
  - Basically, these certificates can be used to identify a person responsible for an unattended service or process acting as client and/or server.
• In order to strongly reduce the risks to have the robot certificate compromised, the INFN CA decided to store this new certificate on board of the Aladdin eToken smart cards;

• The Aladdin eToken smart card can support many certificates;

• A token PIN is prompted every time the user needs to interact with the smart card;
Installation & Configuration

• The [Mkproxy-rhel4.tar.gz](#) tarball contains all the required binaries for RHEL4 compatible platforms.

After unpacking the tarball, copy over the files to their respective locations:

```
cp -rp etoken/bin/* /usr/local/bin
cp -rp etoken/lib/* /usr/local/lib
cp -rp etoken/etc/openssl.cnf /usr/local/etc
```
Installation & Configuration

- Edit the `/usr/local/mkproxy` script and change the `PKCS11_MOD` environment variable

```
export LD_LIBRARY_PATH="$MYDIR/lib:$LD_LIBRARY_PATH"
export PKCS11_ENG="$MYDIR/lib/engine_pkcs11.so"
#export PKCS11_MOD="$MYDIR/lib/libetpkcs11.so"
export PKCS11_MOD="/usr/lib/libetpkcs11.so"
if [ ! -r "$PKCS11_MOD" ];
  then
    export PKCS11_MOD="/usr/local/lib/libetpkcs11.so"
fi
export OPENSSL="$MYDIR/bin/openssl"
export OPENSSL_CONF="$MYDIR/etc/openssl.cnf"
```

- The `mkproxy` script has been tested on:
  - Windows XP (using cygwin) / Vista / 7
  - Linux Fedora Core 5, 8, 9, 11, 12
  - Linux CentOS 4, 5
  - Scientific Linux 4 and 5
  - Linux OpenSuse 10.1, 11.0, 11.1
  - MacOS X 10.5 and higher
With a single grid certificate on your eToken we can generate a grid proxy by issuing the **Mkproxy-rhel4.tar.gz** tarball.

```
mkproxy --label="Robot:MrBayes"
Starting Aladdin eToken PRO proxy generation
Found X.509 certificate on eToken:
label: (eTCAPI) Robot:MrBayes – Giuseppe La Rocca's GILDA ID
  id: 39453945373335312d333545442d343031612d384637302d32384636363930363630423

Add VOMS extensions running the command:

```
voms-proxy-init --noregen -voms <VO>
```

**Engine: pkcs11** set.
Signature ok
subject=/C=IT/O=GILDA/L=INFN Catania/CN=Robot:MrBayes – Giuseppe La Rocca/CN=proxy
Getting CA Private Key
PKCS#11 token PIN:
Your proxy is valid until: Sun Feb 24 03:58:09 CEST 2008-02-23
1. ask for a service
2. create a proxy with the robot certificate
3. execute action
4. get output
5. get the results

The XML/Java-based EnginFrame framework (first scenario)
The Users Tracking System

With the following service you can interact with the User(s) Tracking System embedded on the GENIUS Grid Portal.

Select the view type:
- Global view
- Application view
- Session view
- Advanced query

<table>
<thead>
<tr>
<th>#</th>
<th>USER</th>
<th>HOSTNAME</th>
<th>JOBID</th>
<th>TIME STAMP</th>
<th>DETAILS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>laroce</td>
<td>193.206.208.201</td>
<td>2QHoxZx-4cbTn8UeGc_6g</td>
<td>2010-01-29 08:57:11</td>
<td>MrBayes+JST job submission: run_job.jdl</td>
</tr>
<tr>
<td>2</td>
<td>laroce</td>
<td>193.206.208.201</td>
<td>N/A</td>
<td>2010-01-29 08:51:45</td>
<td>Session started by user</td>
</tr>
<tr>
<td>3</td>
<td>demouser</td>
<td>193.206.208.201</td>
<td>UIBIOWFs_XTucSJDT2flRA</td>
<td>2010-01-28 18:05:37</td>
<td>MrBayes+JST job submission: run_job.jdl</td>
</tr>
<tr>
<td>4</td>
<td>nicola</td>
<td>193.206.208.201</td>
<td>N/A</td>
<td>2010-01-28 15:30:14</td>
<td>Session closed by user</td>
</tr>
</tbody>
</table>

Session(s) statistics:
- nicola [3]
- laroce [4]
- demouser [1]

Application(s) statistics:
- nicola [3]
- laroce [3]
- demouser [4]
The Users Tracking System (cont.)

Querying the L&B server `grid-test-53.trigrid.it`

### larocca

<table>
<thead>
<tr>
<th>#</th>
<th>JobID</th>
<th>Running (Time stamp)</th>
<th>Done (Time stamp)</th>
<th>CPU Time (*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HVTeoADS0nZoZ4fq1_g</td>
<td>2010-03-17 11:52:19</td>
<td>2010-03-17 11:53:26</td>
<td>67</td>
</tr>
<tr>
<td>2</td>
<td>CV__GZ9AcHroJTdxrlqxAg</td>
<td>2010-03-24 15:09:00</td>
<td>2010-03-24 15:19:43</td>
<td>643</td>
</tr>
<tr>
<td>3</td>
<td>hRz7rNmNsra2IekFYz-F5g</td>
<td>2010-03-24 15:14:01</td>
<td>2010-03-24 15:34:43</td>
<td>1242</td>
</tr>
</tbody>
</table>

### demouser

<table>
<thead>
<tr>
<th>#</th>
<th>JobID</th>
<th>Running (Time stamp)</th>
<th>Done (Time stamp)</th>
<th>CPU Time (*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8JeUOM4nxDZcpGcK0yZDGQ</td>
<td>2010-03-17 11:51:37</td>
<td>2010-03-17 11:52:07</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>9WUpa4oUm4gJazSHK1UKeA</td>
<td>2010-03-17 17:32:36</td>
<td>2010-03-17 17:48:18</td>
<td>942</td>
</tr>
<tr>
<td>3</td>
<td>9bBvamtDXXTWXn0_H_s6OQ</td>
<td>2010-03-16 10:47:11</td>
<td>2010-03-16 11:02:49</td>
<td>938</td>
</tr>
</tbody>
</table>

### nicola

<table>
<thead>
<tr>
<th>#</th>
<th>JobID</th>
<th>Running (Time stamp)</th>
<th>Done (Time stamp)</th>
<th>CPU Time (*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UHpzlbQoblFZS30-WlpfeQ</td>
<td>2010-03-16 11:32:45</td>
<td>2010-03-16 11:47:57</td>
<td>912</td>
</tr>
</tbody>
</table>
Pros and Cons of this implementation

• Easy access to the computing resources of the Grid.

• If something is compromised, removing the smart card from the portal the Grid access is based on standard X.509 user’s certificate.

• We need to hack for wrapping Mkproxy-rhel4.tar.gz script in our Grid portals/Science Gateways.

• The solution is centralized!
  – Only one configured server can exploit this authentication mechanism.

The design of Java APIs for supporting a new crypto library and enable a new Grid authentication process based on the use of smart cards is an alternative to resolve these issues!
• Introduction to the “lightweight” crypto library:
  – Java™ PKCS#11, Bouncy Castle and Java CoG Kits;
  – The Architecture;
  – The list of software packages;
  – Examples;
  – Future works.
The Cryptographic Token Interface Standard (PKCS#11)

- The **Cryptographic Token Interface Standard (PKCS#11)** is a standard introduced by **RSA Data Security Inc**;
  
  - It defines native programming interfaces to **cryptographic tokens**, (hardware cryptographic accelerators, smart cards, ...);

- To make easier the integration of these PKCS#11 tokens, the **PKCS#11 provider** has been introduced. The PKCS#11 provider is supported on several platforms;

- PKCS#11 standard includes **sixty** function prototypes (also referred to as **cryptoki library**) that together can be used to perform a wide range of cryptographic operations.
The Bouncy Castle APIs

- The Bouncy Castle APIs provide support for creating two kinds of X.509 certificates:
  - version 1
    - They are used to create root certificates;
      - org.bouncycastle.x509.X509V1CertificateGenerator
  - version 3
    - They contain certificate extensions;
      - org.bouncycastle.x509.X509V3CertificateGenerator
  - PKCS10 certification requests
    - org.bouncycastle.jce.PKCS10CertificationRequest
The Java CoG Kits

• CoG Kits allow users to provide Globus Toolkit functionality within their code without calling scripts, or in some cases without having Globus installed.
  – CoGs are currently available for Java, Python, CORBA, Perl, and Matlab.

• The Java CoG Kits distributed under the Globus Toolkit Public License (GTPL) is an extension of the Java libraries and classes that provides Globus Toolkit functionality.
  – It provides Java classes for interfacing with the following Globus components/functions:
    • Proxy: Credential creation and destruction;
    • GRAM: Job submission and monitoring;
    • MDS: Resource searching;
    • RSL: Resource specification and job execution;
    • GridFTP: Data Management;
    • GASS: Data Management.
The “lightweight” crypto library

- The new “lightweight” crypto library has been designed and developed considering:
  - the native PKCS#11 (v2.0) APIs;
  - the Bouncy Castle APIs;
  - the Cog-jGlobus (ver 1.8.0) APIs;
  - SSL/TLS mechanisms;
  - Java Multithreaded Server.
The new scenario...

- eToken Server manages a list of credentials kept in the smart card.
- TokenClient sends requests for browsing the smart card content and generate VOMS proxies.
- SSL protocol is used to authenticate the server to the client.
The new “lightweight” crypto library includes the following software packages:

- **Java classes:** TokenUtils.java, VOMSUtils.java, MyProxyUtils.java, eTokenServer.java, TokenClient.java, ImportKey.java, VincenzoBase64.java, PasswordField.java

- **Additional libs:**
  - TokenU9ls.java,
  - VOMSU9ls.java,
  - MyProxyU9ls.java,
  - eTokenServer.java,
  - TokenClient.java,
  - ImportKey.java,
  - PasswordField.java,
  - VincenzoBase64.java,
  - PasswordField.java

- **Apache 2.0 license**

- **CoG Kits**

- **VOMSES conf**

- **JKS with the X.509 host certificate of the server**
SSL/TLS encryption

- Apache Tomcat and many Java applications expect to retrieve X.509 certificate from Java Key Store;
  - Convert the certificate and the key into DER format using openssl;
    
    ```bash
    openssl pkcs8 -topk8 -nocrypt -in key.pem -inform PEM -out key.der -outform DER
    openssl x509 -in cert.pem -inform PEM -out cert.der -outform DER
    ```
  - Import these files into the JKS (ImportKey.java)
When **eTokenServer** starts, the VOMS configuration parameters are loaded in a memory HashMap

- The token PIN is provided in input to satisfy the requests of all the authorized clients.
- The **TokenClient** can send to the server encrypted requests for listing the X.509 certificates into the smart card or generating VOMS proxy certificates.
Listing X.509 labels from the eTokenServer

- **When TokenClient sends requests for listing the X.509 labels, the server read all the available credentials stored in the USB token**

```
root@myproxy:--/eTokens-2.0

File Edit View Terminal Tabs Help

Note: eTokenServer.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

<main> DEBUG eTokenServer - Reading VOMSes from ./vomses
<main> DEBUG eTokenServer - Loading VOMS info:
<main> DEBUG eTokenServer - {decide={VOMS_ALIAS=vo.eu-decide.eu, VOMS_SERVER=voms.ct.infn.it, VOMS_HOSTDN=/C=IT/O=INFN/OU=Host/L=Catania/CN=voms.ct.infn.it, VOMS_PORT=15805, VOMS_NAME=decide, VOMS_FILE=./vomses/vo.eu-decide.eu-voms.ct.infn.it}, gilda={VOMS_ALIAS=gilda, VOMS_SERVER=voms.ct.infn.it, VOMS_HOSTDN=/C=IT/O=INFN/OU=Host/L=Catania/CN=voms.ct.infn.it, VOMS_PORT=15001, VOMS_NAME=gilda, VOMS_FILE=./vomses/gilda-voms.ct.infn.it}, eumed={VOMS_ALIAS=eumed, VOMS_SERVER=voms2.cnaf.infn.it, VOMS_HOSTDN=/C=IT/O=INFN/OU=Host/L=CNAF/CN=voms2.cnaf.infn.it, VOMS_PORT=15816, VOMS_NAME=eumed, VOMS_FILE=./vomses/eumed-voms2.cnaf.infn.it}, ops={VOMS_ALIAS=ops, VOMS_SERVER=lcg-voms.cern.ch, VOMS_HOSTDN=/DC=ch/DC=cern/OU=computers/CN=lcg-voms.cern.ch, VOMS_PORT=15809, VOMS_NAME=ops, VOMS_FILE=./vomses/ops-lcg-voms.cern.ch}}

Insert the PKCS#11 token PIN:***********
<main> DEBUG eTokenServer - eTokenServer listening on port 8881
[Thread-2] DEBUG eTokenServer -
[Thread-3] DEBUG eTokenServer - Arguments: {COMMAND=list}
[Thread-3] DEBUG eTokenServer - ACTION~TO~BE~PROCESSED => *** [LISTING Token] ***
[Thread-3] DEBUG TokenUtils - Smart Card: Aladdin eToken PRO 32K 4.2B
[Thread-3] DEBUG TokenUtils - Provider Name: SunPKCS11-eToken
[Thread-3] DEBUG TokenUtils - Version: 1.6
[Thread-3] DEBUG TokenUtils - Size: 35
[Thread-3] DEBUG TokenUtils - 2 Key item(s) detected:
[Thread-3] DEBUG TokenUtils - (eTCAPI) MrBayes's GILDA ID
[Thread-3] DEBUG TokenUtils - (eTCAPI) Giuseppe La Rocca's INFN ID
```

EGI-InSPIRE RI-261323 www.egi.eu
• The **TokenClient** retrieves from the server, the list of available labels (if any)

```bash
[larocca@glite-tutor:/TokenAPIs/TokenClients-2.0]$ ./TokenClient.sh list
Note: TokenClient.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
[main] DEBUG TokenClient - Sending data:
[main] DEBUG TokenClient - {COMMAND=list}
[main] DEBUG TokenClient - Reading list results:
[(eTCAPI) MrBayes's GILDA ID, (eTCAPI) Giuseppe La Rocca's INFN ID]
[larocca@glite-tutor:/TokenAPIs/TokenClients-2.0]$ 
```
TokenClient: create a VOMS proxy

- When TokenClient sends to the eTokenServer a request to create a proxy, the server performs the following steps:
  - Reads X.509 certificate from the smart card and generates a plain proxy for a given token label;
  - Stores a long-term proxy certificate in the MyProxy Server.

```plaintext
root@myproxy:~/eTokens-2.0

* [main] DEBUG eTokenServer - eTokenServer listening on port 8881
[Thread-2] DEBUG eTokenServer -
[Thread-3] DEBUG eTokenServer - Arguments: {PROXY_FILE=/tmp/x509up_u8, MYPROXY_PORT=7512, COMMAND=create, VOMS_FQAN=[eumed:/eumed/Role=SoftwareManager], PROXY_TYPE=GOBIUS, PROXY_KEYBIT=1024, MYPROXY_SERVER=myproxy.ct.infn.it, TOKEN_LABEL=Rocca, PROXY_LIFETIME=12, MYPROXY_LIFETIME=604800}
[Thread-3] DEBUG eTokenServer -
[Thread-3] DEBUG eTokenServer -
[Thread-3] DEBUG eTokenServer - {PROXY_FILE=/tmp/x509up_u8, MYPROXY_PORT=7512, TOKEN_CONFIG_FILE=/eToken.cfg, COMMAND=create, VOMS_FQAN=[eumed:/eumed/Role=SoftwareManager], PROXY_TYPE=GOBIUS, PROXY_KEYBIT=1024, MYPROXY_SERVER=myproxy.ct.infn.it, TOKEN_LABEL=Rocca, PROXY_LIFETIME=12, MYPROXY_LIFETIME=604800}
[Thread-3] DEBUG MyProxyUtils -
[Thread-3] DEBUG MyProxyUtils - Reading proxy from '/tmp/x509up_2876791391952399419.long'
[Thread-3] DEBUG MyProxyUtils -
[Thread-3] DEBUG MyProxyUtils - A proxy valid for user /C=IT/O=INFN/OU=Personal Certificate/L=Catania/CN=Giuseppe La Rocca now exists on myproxy.ct.infn.it
[Thread-3] DEBUG MyProxyUtils - Remaining lifetime: 167 hours (6 days)
[Thread-3] DEBUG eTokenServer -
[Thread-3] DEBUG eTokenServer -
```

EGI-InSPIRE RI-261323 www.egi.eu
Contacts the VOMS Server and adds the VOMS AC to the plain proxy.

TokenClient: create VOMS proxy (cont.)

- Action to be processed => *** [Adding AC Attributes] ***
The standard VOMS proxy is finally sent back to the client.
The beta version of these lightweight Java APIs has been successfully used by the new e-Collaboration environment based on the high customizable features of Liferay portal and the Java/XML EnginFrame 2010 framework.

Future plans

• Integrate the new crypto library in the DECIDE Science Gateway

See Roberto Barbera’s talk – Biomedicine & Life Sciences Applications (I) – 12:30 ~ 14:00
Summary & Conclusions

• The valuable benefits introduced by robot certificates in e-Science can be extended to users belonging to different scientific domains, providing an asset in raising Grid awareness in a wider number of potential users;

• The Java SE platform provides developers with a large set of security APIs, algorithms, tools and protocols;

• We have extended the PKCS#11 cryptographic standard together with the Bouncy Castle and Java CoG Kits APIs to implement a new security solution for the gLite Grid middleware;

• The solution described in this paper can be used by users, applications, Grid portals and/or Science Gateways to generate VOMS proxies starting from the credentials stored on an eToken smart card.
Any questions, comments or remarks are very welcome.

contact: giuseppe.larocca@ct.infn.it