Grid Security Infrastructure

Workshop on Porting Scientific Applications on Computational GRIDs

Riccardo Murri
<riccardo.murri@ictp.it>

EGRID project, The Abdus Salam ICTP, Trieste

Feb 6, 2005

Outline

1. Security in general
   - Public-key cryptography
   - X.509 Public Key Infrastructure
2. Authentication
   - User authentication with GSI
3. Authorization
   - VOMS
   - Authorization schemes
4. References

Facets of security

Authentication	prove the identity of an entity (user, host, service, ...)
Authorization	an entity can do only what it is allowed to
Confidentiality	a third party cannot understand the communication
Integrity	data is not modified during communication
Non-repudiation	it can be verified that the sender and receiver were, in fact, the parties who claimed to send or receive the message

Security in the Grid

- **Authentication**
  Users/hosts/services need to *identify* themselves to build trust relations
  - Users do not know where their jobs will be executed
  - Resource providers do not know the users that will be using them
- **Authorization**
  Restrictions may be imposed on the actions allowed to an entity
  - a normal user may only run software
  - an administrator may also install new software
- **Confidentiality, Integrity, Non-repudiation**
  Requisites of secure computer network communication

Public-key cryptography

> *Public key cryptography is a form of cryptography which generally allows users to communicate securely without having prior access to a shared secret key, by using a pair of cryptographic keys, designated as public key and private key — Wikipedia*

Public and private keys are a pair of transformations (P, P^{-1}), one inverse to the other, such that:

- it is computationally *hard* to find P^{-1}, given P.
- it is computationally *easy* to generate the pair (P, P^{-1})

Public-key cryptography can be used to ensure *Confidentiality*, *Integrity* and *Non-repudiation*.

Confidentiality

Alice wants to send message M to Bob

1. Alice encodes M with Bob's *public* key: $B(M)$
2. Alice sends the encrypted message $B(M)$ over the net
3. Bob decodes the received message with the *private* key: $B^{-1}(B(M)) = M$

No one can decode the encrypted message $B(M)$ without knowing Bob's private key B^{-1}.

	Public key	Private key
Alice	A	A^{-1}
Bob	B	B^{-1}

Verification of origin

Bob wants to be sure that a message has really been sent by Alice

1. Alice encodes M with *private* key:
2. Alice sends encyphered message $A^{-1}(M)$ to Bob
3. Bob decodes $A^{-1}(M)$ with Alice's *public* key: $A(A^{-1}(M)) = M$

This schema ensures *non-repudiation*: Alice cannot claim $A^{-1}(M)$ does not come from her (only Alice knows A^{-1})

	Public key	Private key
Alice	A	A^{-1}
Bob	B	B^{-1}

Cryptographic hash functions

A *cryptographic hash function* is a map H into a fixed finite set (e.g., $0 \ldots 2^N$) that is:

1. *Preimage resistant*: given h it should be hard to find any m such that $h = H(m)$.
2. *Second preimage resistant*: given an input m_1, it should be hard to find another input, m_2 (not equal to m_1) such that $H(m_1) = H(m_2)$.
3. *Collision-resistant*: it should be hard to find two different messages m_1 and m_2 such that $H(m_1) = H(m_2)$.
4. *Efficiently computable*

Famous cryptohashes include: MD5, RIPEMD-160, SHA-1

Digital signature

1. Alice calculates hash h of message m
2. Alice sends $(m, A^{-1}(h))$ to Bob
3. Bob verifies that the hash part $A^{-1}(h)$ is authentic by decyphering it with Alice's public key A
4. Bob verifies the message integrity by comparing the hash of the received message to the locally-computed hash of message m

This schema ensures

- *integrity*: if a cryptographically secure hash is used, an attacker cannot alter *both* message and signed hash.
- *non-repudiation*: origin can be verified

What is still to be solved?

- Who guarantees that Alice's public key is really Alice's public key and not someone else's? *(Authentication)*
- Who guarantees that Alice's private key is known to Alice *only*?

Digital certificates and Certification Authorities

A *digital certificate* associates a user's identity with a public key.
A third party (*Certification Authority*) guarantees that the contents of a digital certificate are correct.

- CAs *sign* digital certificates, to guarantee they are valid;
- all parties that know the CA public key can verify the signature.

To be useful for digital signature and all other crytographic purposes, certificates are generated together with a *private* key.

- but the CA will *not* sign or even see the private key
- private key is *protected* with a passphrase

Also hosts, services, etc. can be certified.

What's in a X.509 certificate?

Type this to display informations on your Grid certificate

grid-cert-info

```
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 2149 (0x865)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: C=IT, O=INFN, CN=INFN Certification Authority
    Validity
      Not Before: Jun 22 14:37:48 2004 GMT
      Not After : Jun 22 14:37:48 2005 GMT
    Subject: C=IT, O=INFN, OU=Personal Certificate, L=ICTP, CN=Riccardo Murri
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (2048 bit)
          Modulus (2048 bit):
              00:bd:18:e6:93:9a:b1:4f:41:f7:ef:8c:60:fc:be:
              d3:d9:e6:2e:72:7a:9e:42:6d:7a:11:45:90:5e:e5:
              ...
  ...
    Signature Algorithm: md5WithRSAEncryption
      8c:bd:d2:8f:b9:60:96:19:3b:4c:9d:18:91:55:74:ee:98:69:
      6a:1d:52:b7:74:12:cf:9b:fa:50:a3:de:2d:e1:74:cc:52:d2:
      ...
```

What's in a X.509 certificate?

```
csh% grid-cert-info
Certificate:
  Data:
    Version: 3 (0x2)
```

A serial number, unique within certificates signed from the same CA

```
Serial Number: 2149 (0x865)
```

```
    Signature Algorithm: md5WithRSAEncryption
    Issuer: C=IT, O=INFN, CN=INFN Certification Authority
    Validity
      Not Before: Jun 22 14:37:48 2004 GMT
      Not After : Jun 22 14:37:48 2005 GMT
    Subject: C=IT, O=INFN, OU=Personal Certificate, L=ICTP, CN=Riccardo Murri
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (2048 bit)
          Modulus (2048 bit):
              00:bd:18:e6:93:9a:b1:4f:41:f7:ef:8c:60:fc:be:
              d3:d9:e6:2e:72:7a:9e:42:6d:7a:11:45:90:5e:e5:
              ...
  ...
    Signature Algorithm: md5WithRSAEncryption
      8c:bd:d2:8f:b9:60:96:19:3b:4c:9d:18:91:55:74:ee:98:69:
      6a:1d:52:b7:74:12:cf:9b:fa:50:a3:de:2d:e1:74:cc:52:d2:
      ...
```

What's in a X.509 certificate?

```
csh% grid-cert-info
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 2149 (0x865)
    Signature Algorithm: md5WithRSAEncryption
```

Issuer DN: identity of the signing CA

```
Issuer: C=IT, O=INFN, CN=INFN Certification Authority
```

```
    Validity
      Not Before: Jun 22 14:37:48 2004 GMT
      Not After : Jun 22 14:37:48 2005 GMT
    Subject: C=IT, O=INFN, OU=Personal Certificate, L=ICTP, CN=Riccardo Murri
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (2048 bit)
          Modulus (2048 bit):
              00:bd:18:e6:93:9a:b1:4f:41:f7:ef:8c:60:fc:be:
              d3:d9:e6:2e:72:7a:9e:42:6d:7a:11:45:90:5e:e5:
              ...
  ...
    Signature Algorithm: md5WithRSAEncryption
      8c:bd:d2:8f:b9:60:96:19:3b:4c:9d:18:91:55:74:ee:98:69:
      6a:1d:52:b7:74:12:cf:9b:fa:50:a3:de:2d:e1:74:cc:52:d2:
      ...
```

What's in a X.509 certificate?

```
csh% grid-cert-info
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 2149 (0x865)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: C=IT, O=INFN, CN=INFN Certification Authority
```

Time validity interval

```
Validity
    Not Before: Jun 22 14:37:48 2004 GMT
    Not After : Jun 22 14:37:48 2005 GMT
```

```
    Subject: C=IT, O=INFN, OU=Personal Certificate, L=ICTP, CN=Riccardo Murri
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (2048 bit)
          Modulus (2048 bit):
              00:bd:18:e6:93:9a:b1:4f:41:f7:ef:8c:60:fc:be:
              d3:d9:e6:2e:72:7a:9e:42:6d:7a:11:45:90:5e:e5:
              ...
  ...
    Signature Algorithm: md5WithRSAEncryption
      8c:bd:d2:8f:b9:60:96:19:3b:4c:9d:18:91:55:74:ee:98:69:
      6a:1d:52:b7:74:12:cf:9b:fa:50:a3:de:2d:e1:74:cc:52:d2:
      ...
```

What's in a X.509 certificate?

```
csh% grid-cert-info
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 2149 (0x865)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: C=IT, O=INFN, CN=INFN Certification Authority
    Validity
      Not Before: Jun 22 14:37:48 2004 GMT
      Not After : Jun 22 14:37:48 2005 GMT
```

Subject DN: identity of the owner

```
Subject: C=IT, O=INFN, OU=Personal Certificate, L=ICTP, CN=Riccardo Mur
```

```
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (2048 bit)
          Modulus (2048 bit):
              00:bd:18:e6:93:9a:b1:4f:41:f7:ef:8c:60:fc:be:
              d3:d9:e6:2e:72:7a:9e:42:6d:7a:11:45:90:5e:e5:
              ...
  ...
    Signature Algorithm: md5WithRSAEncryption
      8c:bd:d2:8f:b9:60:96:19:3b:4c:9d:18:91:55:74:ee:98:69:
      6a:1d:52:b7:74:12:cf:9b:fa:50:a3:de:2d:e1:74:cc:52:d2:
      ...
```

What's in a X.509 certificate?

```
csh% grid-cert-info
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 2149 (0x865)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: C=IT, O=INFN, CN=INFN Certification Authority
    Validity
      Not Before: Jun 22 14:37:48 2004 GMT
      Not After : Jun 22 14:37:48 2005 GMT
    Subject: C=IT, O=INFN, OU=Personal Certificate, L=ICTP, CN=Riccardo Murri
```

Owner's public key (binary data)

```
Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (2048 bit)
          Modulus (2048 bit):
              00:bd:18:e6:93:9a:b1:4f:41:f7:ef:8c:60:fc:be:
              d3:d9:e6:2e:72:7a:9e:42:6d:7a:11:45:90:5e:e5:
              ...
```

```
  ...
    Signature Algorithm: md5WithRSAEncryption
      8c:bd:d2:8f:b9:60:96:19:3b:4c:9d:18:91:55:74:ee:98:69:
      6a:1d:52:b7:74:12:cf:9b:fa:50:a3:de:2d:e1:74:cc:52:d2:
      ...
```

What's in a X.509 certificate?

```
csh% grid-cert-info
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 2149 (0x865)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: C=IT, O=INFN, CN=INFN Certification Authority
    Validity
      Not Before: Jun 22 14:37:48 2004 GMT
      Not After : Jun 22 14:37:48 2005 GMT
    Subject: C=IT, O=INFN, OU=Personal Certificate, L=ICTP, CN=Riccardo Murri
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (2048 bit)
          Modulus (2048 bit):
              00:bd:18:e6:93:9a:b1:4f:41:f7:ef:8c:60:fc:be:
              d3:d9:e6:2e:72:7a:9e:42:6d:7a:11:45:90:5e:e5:
              ...
  ...
```

Digital signature from the CA (binary data)

```
Signature Algorithm: md5WithRSAEncryption
    8c:bd:d2:8f:b9:60:96:19:3b:4c:9d:18:91:55:74:ee:98:69:
    6a:1d:52:b7:74:12:cf:9b:fa:50:a3:de:2d:e1:74:cc:52:d2:
    ...
```

Certificate chains

A CA has its own certificate, signed by another CA

- the verification of a user certificate requires verification of all the steps in the chain
- tree of CAs, end-entities (users, host, etc.) are leaves

A CA can self-sign its certificate

- this is called a "root CA"
- Root CA certificates are usually distributed with software (web browsers, MUAs, etc.)
- widespread adoption is currently the sole barrier against root CA forging

Certificate revocation

CAs publish *Certificate Revocation Lists* (CRL).

- List certificates that should no longer be considered valid, even if still in their validity time
  - Private key compromised
  - User/host lost requisites for certification
  - ...
- new versions published at *fixed* intervals
- download from the web

OCSP: *Online Certificate Status Protocol*

- not yet widely deployed

Grid Security Infrastructure

- GSI is based on an X.509 PKI
- Every user and service involved in the Grid has an X.509 certificate
- Each site chooses which CAs to trust
- Each Grid transaction is mutually authenticated: each party must trust the other parties' CA

User Authentication with GSI

Requisites:

- Single Sign-on: no need to type private key passphrase again and again
- Delegation: jobs and other agents need to act on behalf of the user (with optional restrictions in functionalities)

Problems:

- private key is password-protected
- should not send private key or password over the net

Proxy certificates

Extensions of X.509 digital certificates, defined in RFC 3820.

- user's private key is used to sign a (proxy) digital certificate, composed of a new public/private key pair
- the private key in the proxy is *not* passphrase-protected
- proxy lifetime limited (usually 12 hours) — minimizes risk of "compromised credentials"

The proxy certificates may be sent over the net, with no risk of compromising the user's credentials.

Commands to operate on proxy certificates

grid-proxy-init the "logon to the Grid":

- prompts for the private key passphrase
- creates a proxy certificate

grid-proxy-info If a valid proxy is found, reports subject DN, holder DN, remaining validity time

grid-cert-info Report on subject DN (`-subject` option), validity time, etc.

grid-change-passphrase Change passphrase on *own private key* (not proxy key!)

MyProxy, I

Problem:

- a proxy has limited lifetime (default 12h)
  - bad idea to have a longer one — proxies cannot be revoked
  - however, a grid task may need more time
- user's private key and passphrase is needed to create proxy

MyProxy, II

Solution: the MyProxy server

- allows to create and store a long-term proxy certificate (default 7 days)
- creates short-term proxies from this one at request
- retrieve mode:
  - protects access to the long-term proxy with a password
  - user can retrieve a short-term proxy from any UI
- renew mode:
  - RB can renew proxy if job is still running and proxy expires soon
  - not password protected: only authorized hosts can renew (MyProxy admin chooses authorized hosts)

Note: retrieve and renew mode are not compatible, a proxy created for retrieval may not be renewed and viceversa.

MyProxy commands

Commands to operate the MyProxy service:

myproxy-init	create and store a new long-term proxy -s *hostname* hostname of the MyProxy server to contact
myproxy-info	get information on the stored long-term proxy
myproxy-get-delegation	*retrieve* a new short-term proxy from the server
myproxy-destroy	destroy the long-term proxy on the server

What about authorization?

Job execution

- Access policy is implemented through job queues
- Grid users only need to submit and cancel jobs in a queue
- *Binary* policy: you either can access a queue or you cannot

Data management

- Hierarchical structure of filesystem
- Mixture of access methods: read, write, delete, rename, ...

Globus/GridFTP user mapping

- Map a Grid identity (the subject of a user certificate) to a local account
- Then the Grid user has the same access rights of the local account he is mapped to (file access, disk quotas, CPU limits, etc.)
- The mapping is done via a `grid-mapfile`, which contains a sequence of lines of type:

  "⟨*certificate subject DN*⟩" ⟨*local account*⟩

Problem: is a local account needed for every Grid user?

Pool accounts, I

Pool accounts are a sort of "anonymous" local account for Grid users.

- a set of accounts all belonging to one and the same local UNIX group
- a Grid user is mapped to the first "free" account in the pool
- after some time, the account is "recycled" and ready for assignment to another Grid user
- so you might be refused by a site if there's no more pool accounts free

Pool accounts, II

Only the pool matters: the same Grid user is mapped possibly to a different account in the pool each time it accesses a local machine. Security implications:

- flattening of a hierarchy of access rights — all users in a VO acquire the same rights on the local machine
- cannot protect files per-user, only per-group: pool accounts can access each other's files
- need to be careful in account recycling

What is VOMS?

VOMS embeds additional information on group membership and capabilities (VOMS attributes) in a proxy certificate.

- VOMS attributes are embedded in a *extension* section of the proxy: a VOMS-enabled proxy is a standard proxy to non-VOMS software
- User mapping done on group, role or capability — not on identity alone
- Attributes are digitally signed by the server granting them

VOMS Attributes

A VOMS proxy lists a sequence of attributes.
Each attribute consists of a quadruple:

VO *Virtual Organization* name

Group groups are organized in a tree-like hierarchical structure

Role the set of roles has no hierarchical structure

Capability VO-specific information

The VOMS server

A server managing the Grid user to VOMS attributes mapping.

- one server per VO
- server returns list of attributes a Grid user has access to
- will digitally sign each attribute to ensure validity
- client must know server identity beforehand to be able to verify signature

VOMS commands

voms-proxy-init	create a new VOMS-enabled proxy
	no options — same as `grid-proxy-init`
	-voms *vo* — request VOMS attributes from the VOMS server for the given VO
	-order *group:role* — request only attributes matching given group and/or role
voms-proxy-info	print informations on the (currently active) VOMS proxy
voms-proxy-destroy	destroy the local copy of a VOMS proxy

GridFTP authorization scheme

GridFTP is the most common GSI-enabled server.

- exposes the local filesystem to the Grid
- maps Grid users to local users
- authorization is based on *local user* access rights on the filesystem
    - that is, it behaves just like the plain old FTP daemon

In LCG parlance, a GridFTP host is a “classic SE”.

DPM/LFC authorization scheme

- map user Grid identity (subject DN) to *unique* virtual "User ID" (VUID)
    - VUID bears no relation with the UNIX host UID
- map each VOMS attribute to a *unique* virtual "Group ID" (VGID)
    - VGID bears no relation with the UNIX host GID
- authorization is based on DPM/LFC internal ACLs on data
    - ACLs have POSIX-like semantics, but are not enforced on the filesystem
- authorization info is private to DPM/LFC
    - so you need special versions of the GridFTP, RFIO, etc. daemons

Drawbacks:

- how to share authorization data among different servers?

Public Key Infrastructure

- IETF Public Key Infrastructure Charter: `http://www.ietf.org/html.charters/pkix-charter.html`
  Complete index to PKI-related internet standards and draft standards.
- International Grid Trust Federation: `http://www.gridpma.org/`
  Manages and publishes profiles for Grid CAs, and links to public CA registries.
- Overview of GSI: `http://www.globus.org/security/overview.html`
- Proxy certificates / RFC 3820: `http://www.faqs.org/rfcs/rfc3820.html`

Software

- MyProxy: http://myproxy.ncsa.uiuc.edu/
- VOMS: http://infnforge.cnaf.infn.it/docman/?group_id=7
- DPM and LFC: https://uimon.cern.ch/twiki/bin/view/LCG/DataManagementDocumentation
- Other LCG-2 software: https://uimon.cern.ch/twiki/bin/view/LCG/LCGSoftware
- The LHC Computing Grid, http://lcg.web.cern.ch/LCG/