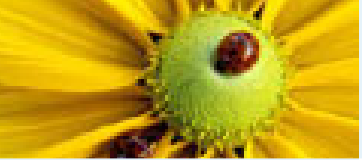

Grid Security Infrastructure

Riccardo Murri
<riccardo.murri@ictp.it>

Nov 16, 2005



Security

Facets of security

Security in the Grid

Public-key
cryptography

X.509 Public Key
Infrastructure

Grid Security
Infrastructure

Authorization

Security



Facets of security

Security

Facets of security

Security in the Grid

Public-key
cryptography

X.509 Public Key
Infrastructure

Grid Security
Infrastructure

Authorization

Five facets of security are of special concern when it comes to Grid computing:

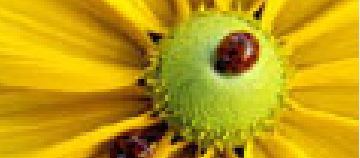
Authentication prove the identity of an entity (user, host, service, ...)

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

Security

Facets of security

Security in the Grid

Public-key
cryptography

X.509 Public Key
Infrastructure

Grid Security
Infrastructure

Authorization

■ 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

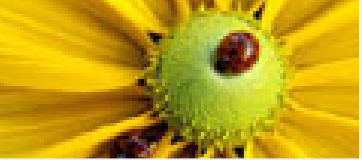
■ Authorization

Restrictions may be imposed on the actions allowed to an entity

- ◆ e.g., a normal user may only run software
- ◆ an administrator may also install new software

■ Confidentiality, Integrity, Non-repudiation

Requisites of computer network communication



Security

Public-key
cryptography

Public-key
cryptography

Confidentiality

Verification of origin

Cryptographic hash
functions

Digital signature

Problems

X.509 Public Key
Infrastructure

Grid Security
Infrastructure

Authorization

Public-key cryptography



Public-key cryptography

Security

Public-key
cryptography

Public-key
cryptography

Confidentiality

Verification of origin

Cryptographic hash
functions

Digital signature

Problems

X.509 Public Key
Infrastructure

Grid Security
Infrastructure

Authorization

“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

Security

Public-key
cryptography

Public-key
cryptography

Confidentiality

Verification of origin

Cryptographic hash
functions

Digital signature

Problems

X.509 Public Key
Infrastructure

Grid Security
Infrastructure

Authorization

Alice wants to send message M to Bob

- Alice encodes M with Bob's *public* key: $B(M)$
- Alice sends the encrypted message $B(M)$ over the net
- 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

Security

Public-key
cryptography

Public-key
cryptography
Confidentiality

Verification of origin

Cryptographic hash
functions

Digital signature

Problems

X.509 Public Key
Infrastructure

Grid Security
Infrastructure

Authorization

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

- Alice encodes M with *private* key:
- Alice sends encyphered message $A^{-1}(M)$ to Bob
- Bob decodes $A^{-1}(M)$ with Alice's *public* key:
 $A(A^{-1}(M)) = M$
 - ◆ need M to be formatted according to some agreed standard to know it was decrypted correctly

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

Security

Public-key
cryptography

Public-key
cryptography

Confidentiality

Verification of origin

Cryptographic hash
functions

Digital signature

Problems

X.509 Public Key
Infrastructure

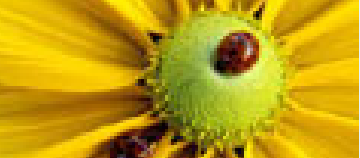
Grid Security
Infrastructure

Authorization

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

- *Preimage resistant*: given h it should be hard to find any m such that $h = H(m)$.
- *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 $H(m_1) = H(m_2)$.
- *Collision-resistant*: it should be hard to find two different messages m_1 and m_2 such that $H(m_1) = H(m_2)$.
- *Efficiently computable*

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



Digital signature

Security

Public-key
cryptography

Public-key
cryptography

Confidentiality

Verification of origin

Cryptographic hash
functions

Digital signature

Problems

X.509 Public Key
Infrastructure

Grid Security
Infrastructure

Authorization

- Alice calculates hash h of message m
- Alice sends $(m, A^{-1}(h))$ to Bob
- Bob verifies that the hash part $A^{-1}(h)$ is authentic by decyphering it with Alice's public key A
- 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



Problems

Security

Public-key
cryptography

Public-key
cryptography

Confidentiality

Verification of origin

Cryptographic hash
functions

Digital signature

Problems

X.509 Public Key
Infrastructure

Grid Security
Infrastructure

Authorization

- 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*?



Security

Public-key
cryptography

**X.509 Public Key
Infrastructure**

X.509 Digital
certificates
What's in a
certificate?

Certificate chains

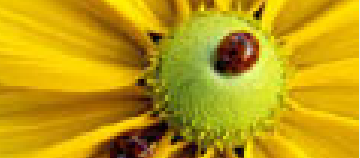
Revocation

GRID@Trieste CA

Grid Security
Infrastructure

Authorization

X.509 Public Key Infrastructure



X.509 Digital certificates

Security

Public-key
cryptography

X.509 Public Key
Infrastructure

X.509 Digital
certificates

What's in a
certificate?

Certificate chains

Revocation

GRID@Trieste CA

Grid Security
Infrastructure

Authorization

A *digital certificate* associates a user's identity with a public key.

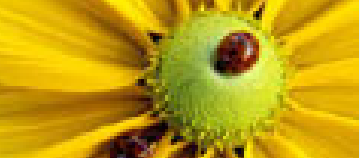
A third party (*Certification Authority*) guarantees that the contents of a digital certificate are correct.

- 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 cryptographic 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 certificate?

Security

Public-key
cryptography

X.509 Public Key
Infrastructure

X.509 Digital
certificates

What's in a
certificate?

Certificate chains

Revocation

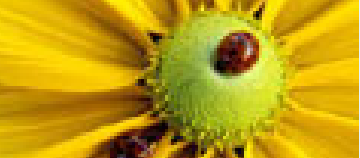
GRID@Trieste CA

Grid Security
Infrastructure

Authorization

An X.509 digital certificate:

- identity of the owner (*subject DN*)
- owner's public key
- validity interval
- identity of the Certification Authority
- digital signature from the CA
- a serial number, that uniquely identifies the certificate among all certificates signed from the same CA



Certificate chains

Security

Public-key
cryptography

X.509 Public Key
Infrastructure

X.509 Digital
certificates
What's in a
certificate?

Certificate chains

Revocation

GRID@Trieste CA

Grid Security
Infrastructure

Authorization

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 the sole barrier against root CA forging



Revocation

Security

Public-key
cryptography

X.509 Public Key
Infrastructure

X.509 Digital
certificates
What's in a
certificate?

Certificate chains

Revocation

GRID@Trieste CA

Grid Security
Infrastructure

Authorization

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
 - ◆ ...
- CA decides how to make public
 - ◆ usually on the web



GRID@Trieste CA

Security

Public-key cryptography

X.509 Public Key Infrastructure

X.509 Digital certificates What's in a certificate?

Certificate chains

Revocation

GRID@Trieste CA

Grid Security Infrastructure

Authorization

- Certifies users and hosts in the GRID@Trieste testbed
 - ◆ one RA per institution/department
 - guarantees for requestor's identity (knows users *personally!*)
 - generates requests for users and forwards them to the CA for signing
 - immediately gives you the *private* key
 - *may* choose to archive your private key for backup purposes
 - ◆ one CA only (at EGRID)
 - signs requests coming from RA
 - mails public key back to requestor and RA
 - archives public certificates at <http://www.egrid.it/gridats/ca/archivio/>



Security

Public-key
cryptography

X.509 Public Key
Infrastructure

**Grid Security
Infrastructure**

GSI: Authentication

GSI

Proxy certificates

Hands on: installing
certificates

Hands on:
`grid-proxy-init`

MyProxy, I

MyProxy, II

Hands on: Myproxy

Authorization

Grid Security Infrastructure



GSI: Authentication

Security

Public-key cryptography

X.509 Public Key Infrastructure

Grid Security Infrastructure

GSI: Authentication

GSI

Proxy certificates

Hands on: installing certificates

Hands on:

`grid-proxy-init`

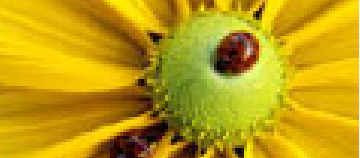
MyProxy, I

MyProxy, II

Hands on: Myproxy

Authorization

- GSI is based on an X.509 PKI
- Every user/host involved in the Grid has an X.509 certificate
- Certificates are signed by trusted CAs
- Each site trusts the CAs it wants
- Each Grid transaction is mutually authenticated: each party must trust the other parties' CA



GSI

Security

Public-key
cryptography

X.509 Public Key
Infrastructure

Grid Security
Infrastructure

GSI: Authentication

GSI

Proxy certificates
Hands on: installing
certificates

Hands on:
grid-proxy-init

MyProxy, I

MyProxy, II

Hands on: Myproxy

Authorization

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

Security

Public-key
cryptography

X.509 Public Key
Infrastructure

Grid Security
Infrastructure

GSI: Authentication
GSI

Proxy certificates

Hands on: installing
certificates

Hands on:
`grid-proxy-init`

MyProxy, I

MyProxy, II

Hands on: Myproxy

Authorization

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
- proxy lifetime limited (usually 12 hours) — minimizes risk of “compromised credentials”
- the private key in the proxy is *not* passphrase-protected

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



Hands on: installing certificates

Security

Public-key
cryptography

X.509 Public Key
Infrastructure

Grid Security
Infrastructure

GSI: Authentication
GSI

Proxy certificates

Hands on: installing
certificates

Hands on:
grid-proxy-init

MyProxy, I

MyProxy, II

Hands on: Myproxy

Authorization

Install the Demo User certificate (you should *not* do this if you already have your own certificate from GRID@Trieste CA).

Copy the certificates into the right location:

```
mkdir .globus
```

```
cp /etc/skel/.globus/usercert.pem .globus/
```

```
cp /etc/skel/.globus/userkey.pem .globus/
```

Ensure the correct permissions:

```
chmod 0644 .globus/usercert.pem
```

```
chmod 0400 .globus/userkey.pem
```



Hands on: grid-proxy-init

Security

Public-key cryptography

X.509 Public Key Infrastructure

Grid Security Infrastructure

GSI: Authentication GSI

Proxy certificates Hands on: installing certificates

Hands on: grid-proxy-init

MyProxy, I

MyProxy, II

Hands on: Myproxy

Authorization

grid-proxy-init is the “logon to the Grid”.

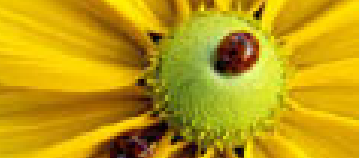
- prompts for the private key passphrase
- creates a proxy certificate

Other proxy-related commands:

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

Security

Public-key
cryptography

X.509 Public Key
Infrastructure

Grid Security
Infrastructure

GSI: Authentication
GSI

Proxy certificates
Hands on: installing
certificates
Hands on:
grid-proxy-init

MyProxy, I

MyProxy, II

Hands on: Myproxy

Authorization

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 longer...
- user's private key and passphrase is needed to create proxy



MyProxy, II

Security

Public-key
cryptography

X.509 Public Key
Infrastructure

Grid Security
Infrastructure

GSI: Authentication
GSI

Proxy certificates
Hands on: installing
certificates
Hands on:
grid-proxy-init
MyProxy, I

MyProxy, II

Hands on: Myproxy

Authorization

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.



Hands on: Myproxy

Security

Public-key
cryptography

X.509 Public Key
Infrastructure

Grid Security
Infrastructure

GSI: Authentication
GSI

Proxy certificates

Hands on: installing
certificates

Hands on:
grid-proxy-init

MyProxy, I

MyProxy, II

Hands on: Myproxy

Authorization

Commands to operate the MyProxy service:

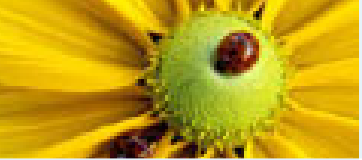
myproxy-init create and store a new long-term proxy

-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



Security

Public-key
cryptography

X.509 Public Key
Infrastructure

Grid Security
Infrastructure

Authorization

User mapping
Pool accounts

Authorization



User mapping

Security

Public-key
cryptography

X.509 Public Key
Infrastructure

Grid Security
Infrastructure

Authorization

User mapping

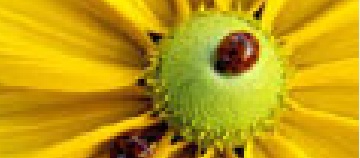
Pool accounts

What is presently done for authorization?

- Map a Grid identity (the subject of a user certificate) to a CE 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

Security

Public-key
cryptography

X.509 Public Key
Infrastructure

Grid Security
Infrastructure

Authorization

User mapping

Pool accounts

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

- a set of accounts all belonging to one and the same (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

Security implications:

- pool accounts can access each other’s files (don’t trust the SE for very sensitive data!)