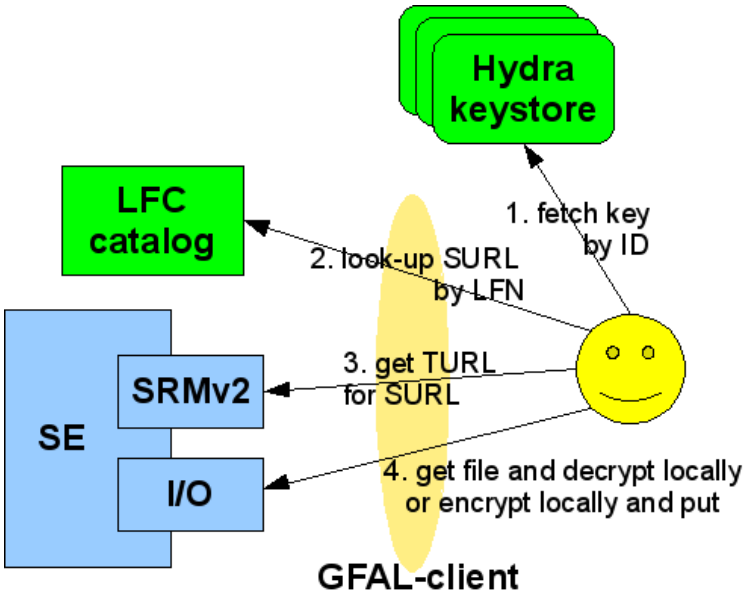
Table of Contents

Encrypted Storage and Hydra

What is Hydra?

Hydra is encrypted file storage solution. It encrypts files and stores them on normal storage elements (storage elements are not part of the Hydra service - any SE supported by GFAL library can be used for that). The sensitive information is the encryption key, which is after generation splitted and distributed on several different places, called the Hydra Keystores (Hydra Servers or services).

The reason for the splitting is that the encryption key can't be reconstructed from the one piece only, i.e. if one of the Hydra server gets compromised, the hacker cannot decrypt the files. Splitting algorithm (SSSS) is however also fault tolerant: the encryption key can be reconstructed if at least some (predefined) count of the pieces can be found (this is normally more than one but less than all). Thus even if one Hydra service does not work, the key can be reconstructed from the rest of the pieces.



Hydra is released in Glite 3.1 Update 38.

Hydra Server

The Hydra server consists of a MySQL database and a Tomcat server which runs the actual hydra software (java). The hyrda server stores only key pieces and related ACL information into MySQL database, not any files.

installation of the service

One has to install at least one Hydra service (glite-data-hydra-service) and must have some storage element. There is separate glite-yaim-hydra package to help configure the Hydra service(s) by YAIM tool. Note! The next message is displayed when running yaim for the first time:

```
ERROR 1045 (28000): Access denied for user 'hydra1'@'localhost' (using password: YES)
```

Message above is a cosmetic warning and can be skipped (bug #43945).

Each server must have MySQL database and Tomcat-server with valid host certificates. Please have a look at

JPackage installation for the basic Java packages.

For test purposes one server can run several hydra services (e.g. 1 MySQL, 3 MySQL databases, 1 Tomcat, 3 webapps).

test configuration of the service

For test purposes one can configure all Hydra service instances on the same node - in production environment this doesn't make sence. You can use this configuration example, where the administrator's DN could be the host itself, what you can get as

```
openssl x509 -in /etc/grid-security/hostcert.pem -subject -noout
```

Configuration file:

```
HYDRA_INSTANCES="1 2 3"
HYDRA_DBNAME_1=hydra_db_1
HYDRA_DBUSER_1=hydra1
HYDRA_DBPASSWORD_1=hydra1
HYDRA_CREATE_1=/dteam/Role=NULL/Capability=NULL
HYDRA_ADMIN_1=<admin-dn>
HYDRA_DBNAME_2=hydra_db_2
HYDRA_DBUSER_2=hydra2
HYDRA_DBPASSWORD_2=hydra2
HYDRA_CREATE_2=/dteam/Role=NULL/Capability=NULL
HYDRA_ADMIN_2=<admin-dn>
HYDRA_DBNAME_3=hydra_db_3
HYDRA_DBUSER_3=hydra3
HYDRA_DBPASSWORD_3=hydra3
HYDRA_CREATE_3=/dteam/Role=NULL/Capability=NULL
HYDRA_ADMIN_3=<admin-dn>
```

You can use this configuration directly with the `/opt/glite/etc/glite-data-hydra-service/configure` command or by appending it to the YAIM configuration node and executing

```
/opt/glite/yaim/bin/yaim -c -s /opt/glite/yaim/etc/site-info.def -n HYDRA
```

If you have installed glite-data-hydra-cli you can test the configuration by

```
source /opt/glite/etc/profile.d/grid-env.sh
glite-eds-key-register testkey
glite-eds-key-unregister testkey
```

The log files of the hydra services are located at
`/var/log/tomcat5/glite-data-hydra-service-<instance>.log`

You can also test the BDII info provider locally:

```
source /opt/glite/etc/profile.d/grid-env.sh
export GLITE_SD_PLUGIN=bdii
export LCG_GFAL_INFOSYS=localhost:2170
glite-sd-query -t org.glite.Metadata
glite-eds-key-register testkey
glite-eds-key-unregister testkey
```

distributed configuration of the service

In a real installation you would have to install the Hydra services on different hosts. If they are on host1.example.org, host2.example.com, host3.example.net, then the confiuguration files would be:

host1.example.org

```
HYDRA_INSTANCES="1"
HYDRA_DBNAME_1=hydra_db
HYDRA_DBUSER_1=hydramgr
HYDRA_DBPASSWORD_1=hydrapwd
HYDRA_CREATE_1=/dteam/Role=NULL/Capability=NULL
HYDRA_ADMIN_1=<admin-dn>

HYDRA_PEERS="2 3"
HYDRA_CREATE_2=/dteam/Role=NULL/Capability=NULL
HYDRA_ID_2=1
HYDRA_HOST_2=host2.example.com
HYDRA_CREATE_3=/dteam/Role=NULL/Capability=NULL
HYDRA_ID_3=1
HYDRA_HOST_3=host3.example.net
```

host2.example.com

```
HYDRA_INSTANCES="1"
HYDRA_DBNAME_1=hydra_db
HYDRA_DBUSER_1=hydramgr
HYDRA_DBPASSWORD_1=hydrapwd
HYDRA_CREATE_1=/dteam/Role=NULL/Capability=NULL
HYDRA_ADMIN_1=<admin-dn>

HYDRA_PEERS="2 3"
HYDRA_CREATE_2=/dteam/Role=NULL/Capability=NULL
HYDRA_ID_2=1
HYDRA_HOST_2=host1.example.org
HYDRA_CREATE_3=/dteam/Role=NULL/Capability=NULL
HYDRA_ID_3=1
HYDRA_HOST_3=host3.example.net
```

host3.example.net

```
HYDRA_INSTANCES="1"
HYDRA_DBNAME_1=hydra_db
HYDRA_DBUSER_1=hydramgr
HYDRA_DBPASSWORD_1=hydrapwd
HYDRA_CREATE_1=/dteam/Role=NULL/Capability=NULL
HYDRA_ADMIN_1=<admin-dn>

HYDRA_PEERS="2 3"
HYDRA_CREATE_2=/dteam/Role=NULL/Capability=NULL
HYDRA_ID_2=1
HYDRA_HOST_2=host1.example.org
HYDRA_CREATE_3=/dteam/Role=NULL/Capability=NULL
HYDRA_ID_3=1
HYDRA_HOST_3=host2.example.com
```

Hydra Client

Hydra client is collection of `glite-eds-*` commands and a hydra-cli library to en/decrypt files and to control hydra keys.

installation of the client

On the UI and WN one needs the client package (glite-data-hydra-cli) with all of its dependencies.

configuration of the client

The client requires some information of the current Hydra servers, this can be a local file (for test purposes) or any information system (e.g. BDII).

`/opt/glite/share/doc/glite-data-hydra-cli/example.services.xml` is a good starting point for test configuration. In this case you should define SD environment variables pointing to your local file

```
export GLITE_SD_PLUGIN=file
export GLITE_SD_SERVICES_XML=/path/to/hydra-services.xml
```

You must also have valid X509 user certificate (environment variables `X509_USER_CERT` and `X509_USER_KEY`).

client usage

One could store a file on the local storage element of the UI:

```
glite-eds-put -i $ID picture rfio:////dpm/cern.ch/home/biomed/mdm/$ID
```

Note! There is a bug (#41592) in the glite-eds-put: you **must** specify the ID with -i option, because the command can't generate the GUID for the file correctly in all cases.

The file could be accessed for local processing:

```
glite-eds-get -i $ID rfio:////dpm/cern.ch/home/biomed/mdm/$ID picture
display picture
```

This copies the file content dencrypted to the local disk, which might not be desirable. One can avoid this step, by directly using the hydra-cli libraries from the application directly. In this case the content is only decrypted in the memory.

Later the user may delete the sensitive file along with its keys:

```
rfio-rm rfio:////dpm/cern.ch/home/biomed/mdm/$ID
glite-eds-rm $ID
```

Usecase: DPM SE with DICOM backend

For DICOM the special DPM server should be installed as back-end.

Solution : Extension of the DPM storage element to support DICOM as storage back-end.

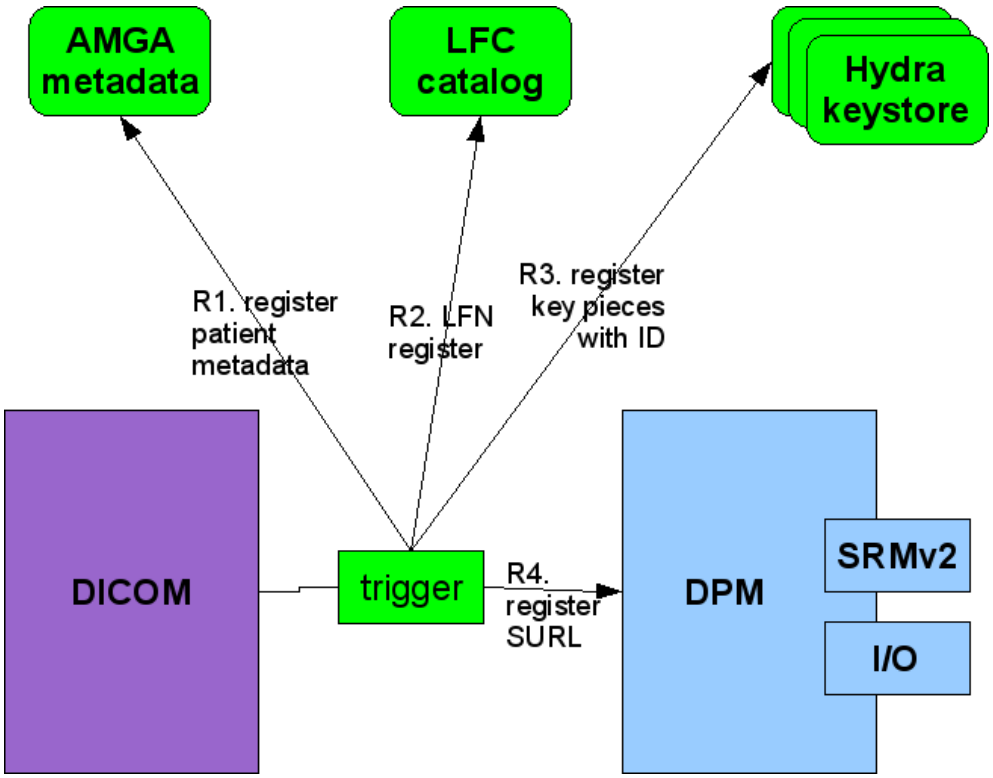
- File encryption on the fly, local decryption
- Use of HYDRA for split key management
- Use of the LFC to register/retrieve system data
- Replicas location, filesize, ...
- Use of srmv2 to get the turls
- Use of block I/O protocols, gridftp to load medical images
- Access control based on VOMS

registration

Using the distributed Hydra keystore

- the file ID=/study/series/SOP ID
- uses DPM with DICOM back-end
- LFN=lfn-prefix/study/series/SOP ID
- SURL=se-prefix/study/series/SOP ID

The back-end is part of a standard DPM service:




```
dpm-dicom-trigger MDM_test-0.78/test/dicom/jm0301-00032.dcm
```

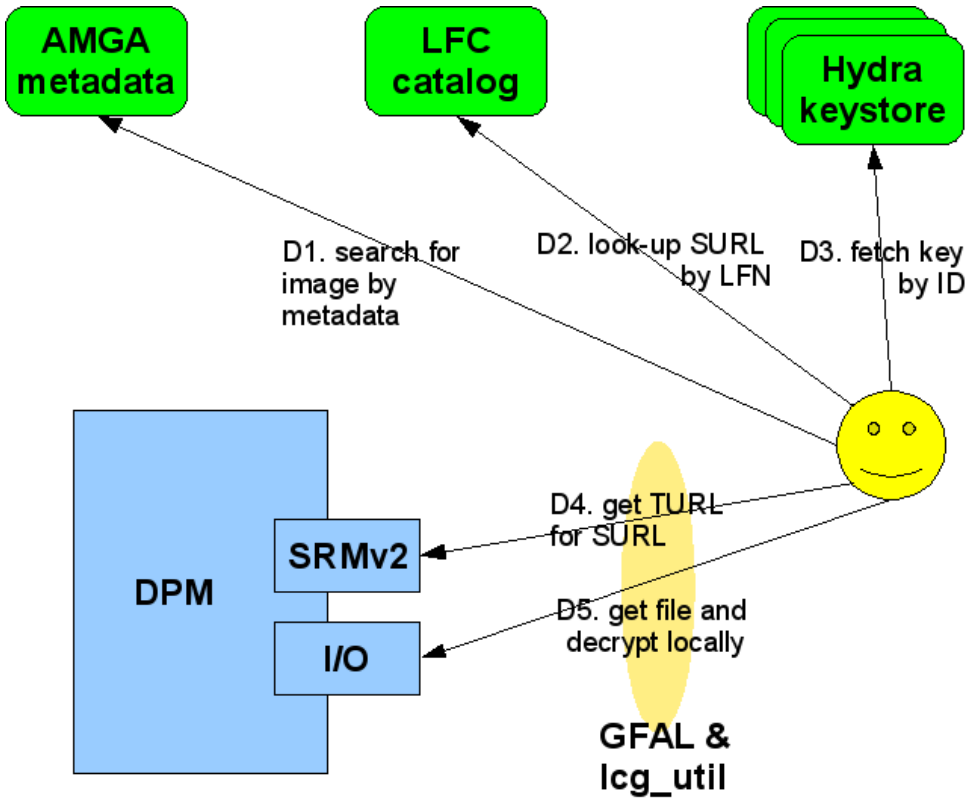
What happens in details:

- uploads a file to DICOM
- fetches the file from DICOM (file size may change!)
- calculates the encrypted size and ID, for example:
- ID=/1.2.826.0.1.3680043.2.1143..20060202124502298.29/1.2.826.0.1.3680043.2.1143..20060202124502298.
- calls 'dpm-register' to register the file into DPM with the SFN /dpm/<domain>/home/biomed/mdm/<ID> and with the PFN dicom:///<ID> and marked as near-line copy
- calls 'glite-eds-register' to create an en/decryption key and adds the DPM host to be able to read it (i.e. glite-eds-setacl)
- optionally: registers the file in LFC with the LDN /grid/biomed/mdm/<ID>
- optionally: registers the file metadata in AMGA

retrieval

The encrypted file can be replicated and retrieved from any other SE.

DPM I/O access via: gridftp, rfio(s), http(s)



Example with separate download an decryption steps:

```
lcg-cp -bD srmv2 srm://dpm.example.org:8446/srm/managerv2?SFN=/dpm/example.org/home/biomed/mdm/
glite-eds-decrypt <ID> picture.enc picture
```

One can do the same with one command:

```
glite-eds-get -i <ID> rfio:////dpm/example.org/home/biomed/mdm/<ID> picture
```

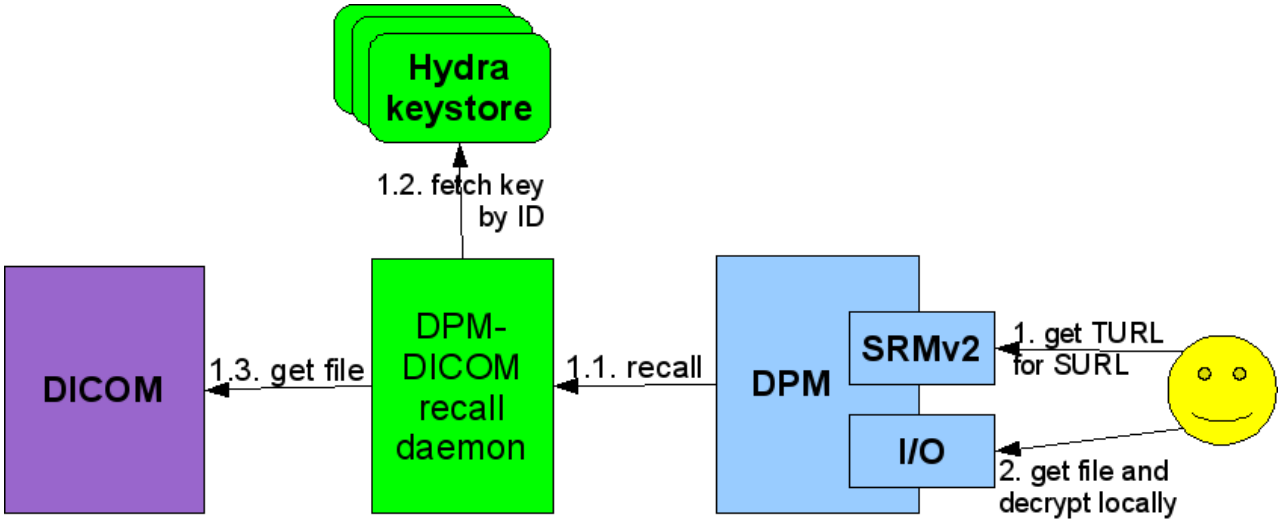
the details in this case:

- file is opened via gfal_open()
- decryption key is fetched for <ID>
- loop on gfal_read(), glite_eds_decrypt_block(), write()

'glite-eds-get' is a simple utility over the EDS library.

internals

DICOM files are marked as 'nearline', which have to be recalled to disk nearline TURL is "study/series/SOP ID" a DPM-DICOM recall daemon processes multi-file recalls from DICOM to disk during the recall the DICOM files are anonymized and encrypted DPM serves the encrypted files normally.



DPM-DICOM recall daemon with multiple multi-file requests: DPM starts the retrieval by a direct call, however jobs are also stored in DB recall daemon manages the DB and DPM interactions and calls a plug-in for each PFN (see dicom:///<ID>) simple plug-in interface, within a shared library in libdpm_dicom.so:

```
int dpm_dicomcopyfile (char *dicom_fn, char *turl, int *errcode, char *errbuf)
```

The plug-in then retrieves the encryption key from Hydra, retrieves the file from DICOM and stores the file via rfio.

Technical background

Old working page of DMEncryptedStorage!

What is SSSS?

Shamir's Secret Sharing Scheme is a library to split keys for reliability and security reasons. The Hydra client library uses SSSS to split and merge encryption keys.

However there is also a simple binary to make use of the library alone:

```
$ glite-ssss-split-passwd -q 5 3 secret
137c9547aba101ef 6ee7adbbaacac1ef 1256bcc160eda592 fdabc259cdfbacc9 3113be83f203d794
$ glite-ssss-join-passwd -q 137c9547aba101ef NULL 1256bcc160eda592 NULL 3113be83f203d794
secret
```

In the hydra client the minimum count (N) of the needed pieces to reconstruct the key is calculated from the total count (M) of the pieces (i.e. number of the servers): `N = ceil(log2( M + 1 ))`. I.e. for 2-3 servers 2 pieces is required, for 4-7 servers 3 pieces is required, etc.

Explanation of key register

HydraEDSKeyRegister

Current Hydra service install commands.

HydraServerInstall

Papers

- Bridging clinical information systems and grid middleware : a Medical Data Manager HealthGrid 2006 conference, 7-9 June 2006, Valencia, Spain, Best Paper award
- Implementation of a Medical Data Manager on top of gLite services PDF
- EGEE user forum (CERN, March 1-3 2006) : Book of Abstracts Encrypted Data Storage in EGEE

Presentations

- EGEE-JRA1 All Hands Meeting, 2006-07 pdf odp
- CHEP Poster, 2007-09 pdf odg
- EGEE JRA1 All Hands Meeting, 2007-10 pdf odp
- Internal MDM Demo, 2007-12 * pdf * odp

Last edit: JohnWhite on 2012-10-26 - 14:19
Number of topics: 1

Maintainer: JoniHahkala, AkosFrohner

This topic: EGEE > DMEDS
Topic revision: r18 - 2012-10-26 - JohnWhite