DIRAC Security Infrastructure

Adrià Casajús
Ricardo Graciani

Overview

- What's DIRAC?
- Security Infrastructure
- What's DISET?
- Basics of DISET
- Authorization and authentication scheme
- DISET portals
- DISET transfers

What's DIRAC?

- Distributed Infrastructure with Remote Agent Control.
  - See talks by Andrei Tsaregorodtsev [301] and Stuart Paterson [260]
- DIRAC is the LHCb interface to the grid for all distributed computing tasks.
- Based on a cooperating services exposing XML-RPC interfaces.
- Mostly coded in Python.

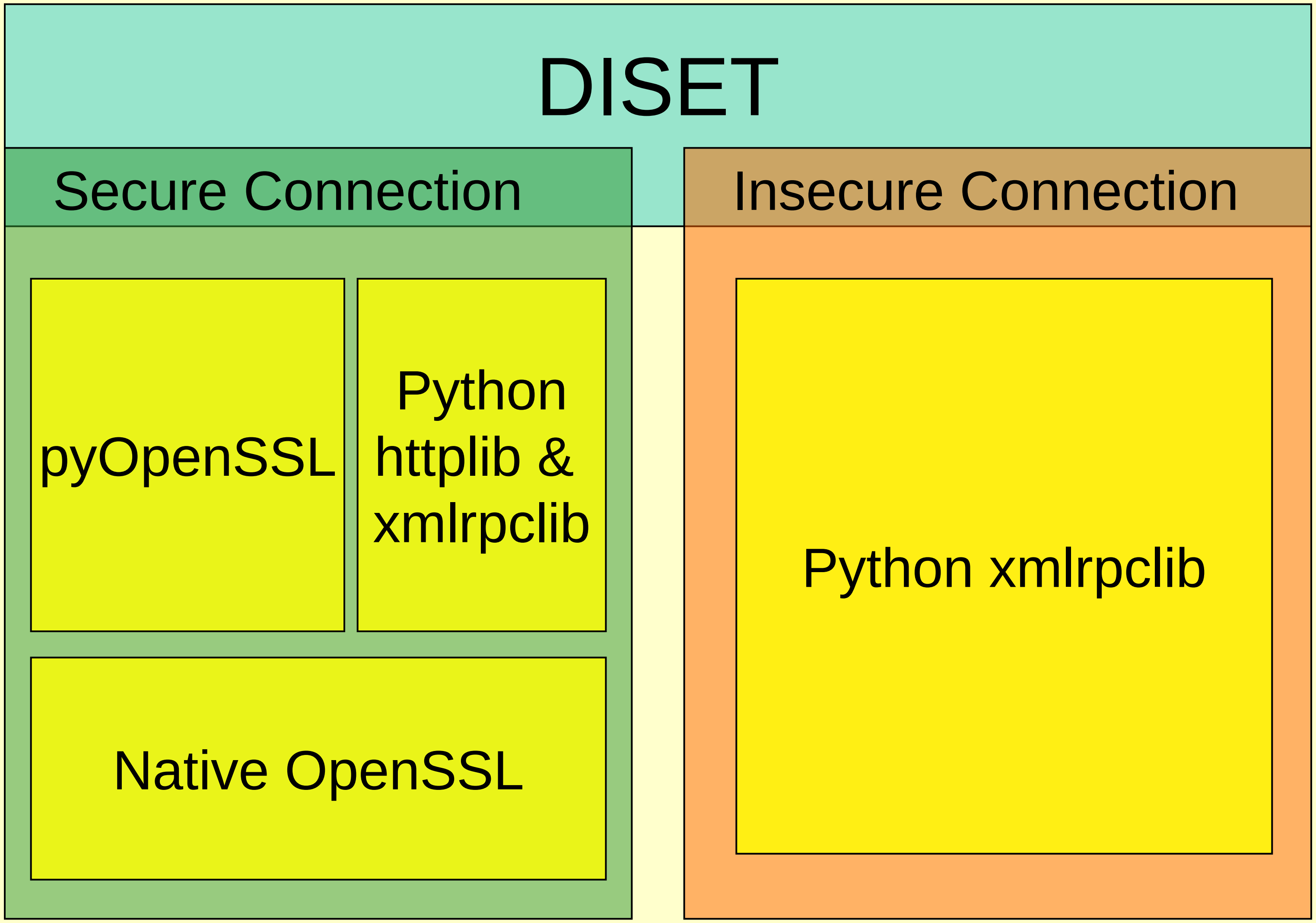
Security Infrastructure

- Based on:
  - Trusted certification authorities (CA's) for authentication.
  - Virtual organizations (VO's) for authorization.
- We wanted to skip globus and minimize dependencies.
- DIRAC applications use grid proxies to connect to services.
  - Based on standard x509 certificates.
- The server has also to be authenticated by clients.

What's DISET?

- DIRAC Secure Transport
- Extension of HTTPS supporting x509 certificates and grid proxies plus enhancement of native XML-RPC capabilities.
- Wraps together all nasty ssl code, authorization and authentication mechanisms under simple calls.
- Performs over 200 queries/second.
  - ~10 times more than other implementations tested.

DISET architecture



DISET dependencies

- Relies on:
  - pyOpenSSL: 3rd party module encapsulating some of the native OpenSSL functionalities.
  - OpenSSL: Open source full-featured toolkit implementing Secure Sockets Layer (SSL v2/3) and Transport Layer Security (TLS)
- OpenSSL handles all underlying authentication except grid proxies.
- Some modifications were made to pyOpenSSL to achieve the needed functionalities.

DISET usage

- DISET client is used exactly the same way as native python XML-RPC one.

```
oSecureClient = DISETClient( “http://lxgate14.cern.ch:9130”)
oSecureClient.get( “SectionName”, “OptionName” )
oInsecureClient = DISETClient( “https://lxgate03.cern.ch:9091” )
oInsecureClient.rescheduleJob( iJobID )
```

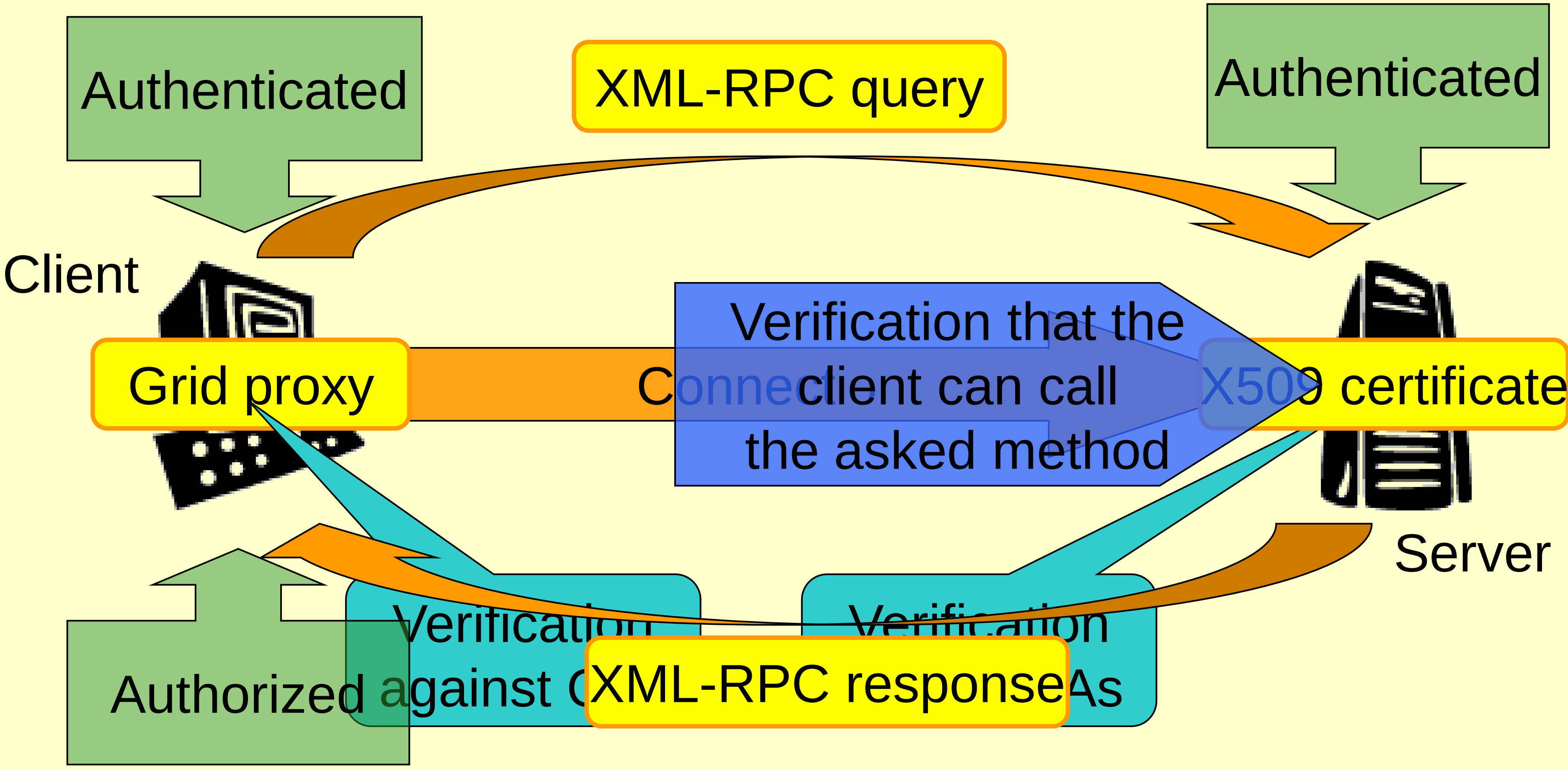
- Changing one line in the server code makes it secure. Server logic remains the same.

```
class FakeHandler( DISETRequestHandler ):
          def export_fakeMethod( self, uSomeParam ):
                    doSomething( uSomeParam )
```

```
oSecureServer = DISETSecureServer( ( “”, iPort ), FakeHandler, “ServiceName”)
oSecureServer.serve_forever()
```

```
oInsecureServer = DISETInsecureServer( ( “”, iPort ), FakeHandler,“ServiceName”)
oInsecureServer.serve_forever()
```

Authorization and authentication scheme



Authorization and authentication scheme

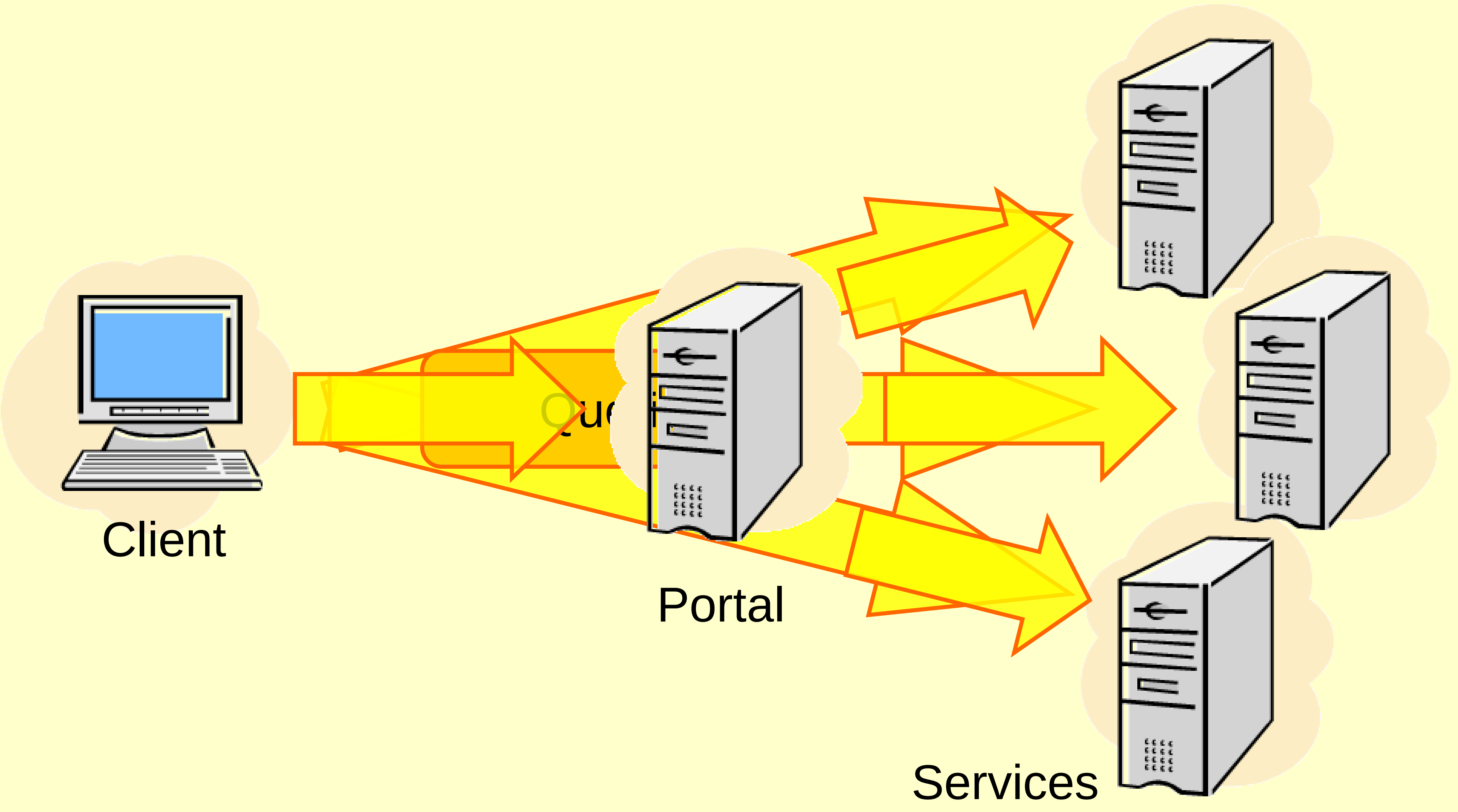
- Authentication:
  - All handled by DISET.
  - Use of ssl sessions for each client. Just one handshake for multiple calls.
    - Session lifetime defined as a server parameter.
- Authorization:
  - Two levels of authorization based on user groups.
  - Users can belong to more than one group, but they have to specify which one they want to use.

Authorization levels

- First level:
  - Automatically checked by DISET based on per method policies exposed in the configuration service.
  - DISET server verifies user's DN to belong to the specified group and the group to be authorized for the method called.
- Second level:
  - Finer grained but optional.
  - Programmed by the developer inside the handler's methods.
  - DISET provides all needed information.

DISET Portals

DISET Portals



DISET Portals

- Destination service is based on the connecting URL.
  - Portal URL + Service Name

    https://PortalLocation/ServiceOne/ → https://ServiceOneLocation/
- Two kind of portals:
  - Secure portals:
    - Programmed in python using DISET.
    - Can redirect to either secure and insecure services.
  - Insecure portals:
    - Can only redirect to insecure services.
    - Available also in PHP + web server.

Advantages of DISET Portals

- Single entry point for all services.
- Allow redirections for load balancing.
- Minimize ssl authentications using sessions:
  - Servers receive handshakes only from portals for all client queries.
  - Clients handshake only once for all queries through the portal.

Secure DISET Portals

- Total transparency for developers:
  - Act as DISET servers for the originating client
  - Act as DISET client for the final server.
- Secure portals need a valid certificate.
- Authentication of the client happens at the portal and the query authorization at the final server.
- Final servers need a list of trusted portals.

DISET Transfers

DISET Transfers

- DISET allows to transfer files from and to servers.
- DISET transfers use the same authentication and authorization mechanism as other DISET methods.
- Transfer request is sent via XML-RPC.
- Once the transfer is accepted, data is sent in binary format through the same connection.

DISET Transfers

- To enable transfers, server developer must code some specific callbacks in the handler:
  - For “put” transfers (client → server) a server needs:
    - allowPutFile(self, sID, sFilename)
    - receiveFile(self, stFileData)
  - For “get” transfer (server → client) a server needs:
    - allowGetFile(self, sID, sFilename)
    - sendFile(self, stFileData)

DISET Example

- Data is sent and received using helper functions.
- Client example:

```
oClient = HSGETransferClient( “https://somewhere:%d” % iPort )
If oClient.putFile( “/etc/motd”, sJobID, “motd” )[ ‘Status’ ] == “Error”:
   processError()
```

- Server example:

```
Class ExampleRH (HSGERequestHandler):
   def allowPutFile( self, sID, sFilename ):
     return S_OK()
   def receiveFile( self, stFileData ):
     sData = “dummy”
     while len( sData ) > 0:
       self.doSomething( sData )
       sData = self._getDataPacket()
   def export_doSomething( self, iSomeParam ):
     return doSomethingWithParam( iSomeParam )

oServer = HSGEServer( ( “”, iPort ), ExampleRH, “ExampleTransfer” )
oServer.serve_forever()
```

Summary

- DISET provides an “easy to use” framework for deploying GSI enabled services.
- Dependencies have been minimized, just OpenSSL is needed to run.
- Securing a connection implies changing one line of code at the server and the service URL at the client.