

# Status of security in XROOTD

- Introduction
- XrdSec
- Status of plug-ins
  - password-based, key-based
  - GSI
- Future plans

# Introduction

- At present only Kerberos can be used to control access to XROOTD servers
- More plug-ins required (password-based, GSI, ...)
- **Direct use of ROOT authentication code** initially considered (straightforward for ROOT Client / Server applications)
- However:
  - **XROOTD is NOT a ROOT application**; the server size could nonetheless be used, though at the expense of a **violation of the XROOTD protocol** (loose control of the network link during handshake)
  - The future of the client (TXNetFile) is to be a wrapper around its son (**XrdClient**) which **is NOT a ROOT application** either.

# Introduction

- ROOT authentication scheme is rather complete, but the evolution of a much less ambitious design:
  - server code **designed for** standalone lightweight daemons **rootd/proofd**
  - **Client / Server asymmetry**, some code duplications, reduced but not eliminated
- Cleaner design envisaged, keeping same features
- **XrdSec natural candidate**: reuse the experience acquired with ROOT, and as much as code as possible, to complete XrdSec

# XrdSec

- C++ framework to manage protocols as plug-ins
- Generic protocol (XrdSecProtocol)

```
class XrdSecProtocol {
public:

virtual int Authenticate( XrdSecCredentials *cred,      // In
                          XrdSecParameters **parms,   // Out
                          XrdSecClientName &client,   // Out
                          XrdOucErrInfo *einfo=0 ) // Out

virtual XrdSecCredentials *getCredentials(
                          XrdSecParameters *parms=0, // In
                          XrdOucErrInfo *einfo=0 ) // Out

virtual const char *getParms( int &psize,             // Out
                              const char *host )      // In
};
```

server

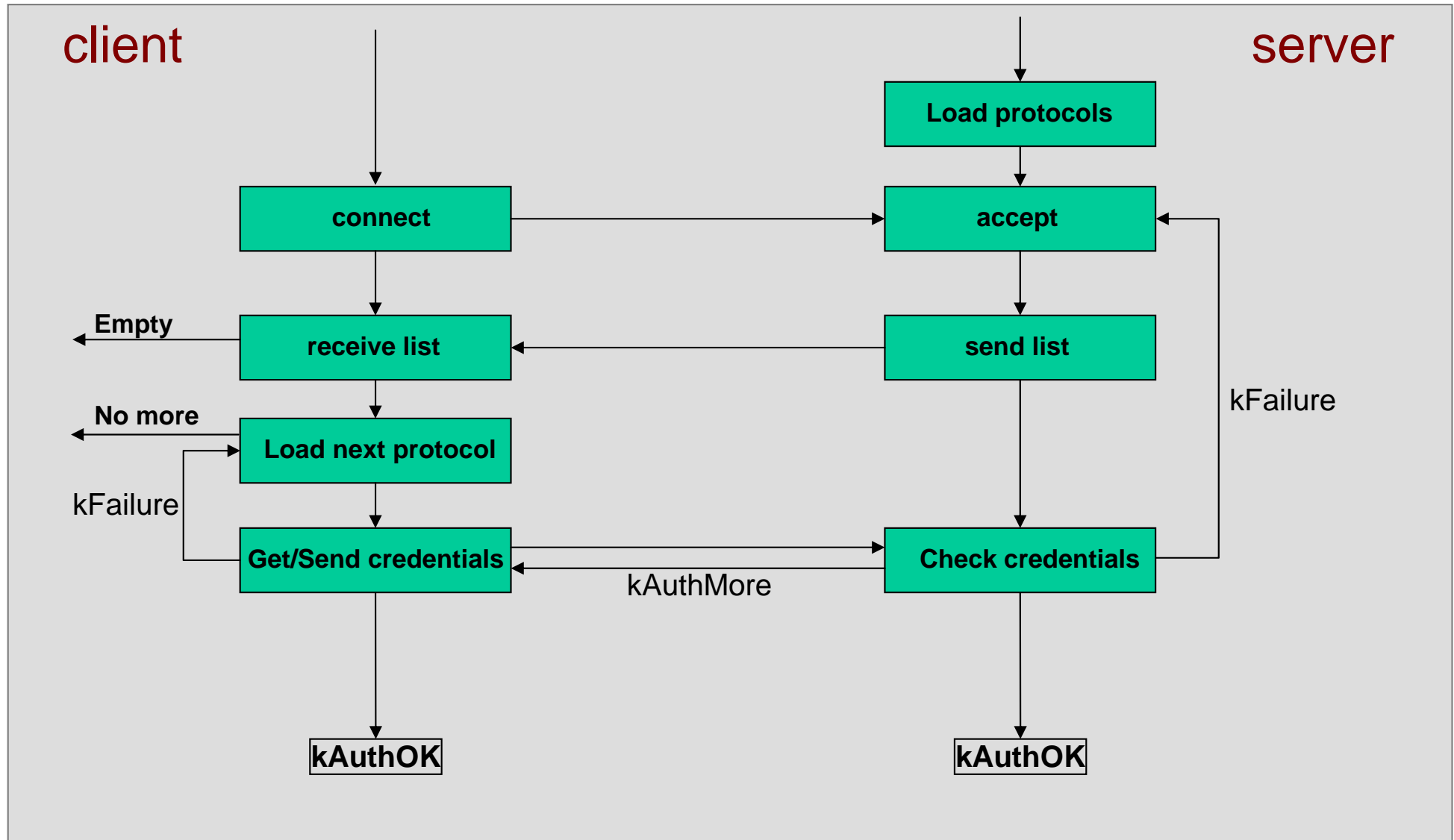
client

- **Protocols** implementations **inherit** from XrdSecProtocol

# XrdSec

- `libXrdSec.so` provides the *Protocol Manager*
  - **Server**: instantiated at start-up from configuration file:
    - load protocol plug-ins that server can / wants to run
    - binds (subsets of) the list to hosts, providing **access control** on host base
  - **Client**: build-up list loading protocols the first time needed
- Plug-in implementations provide a public *instantiator* to create an instance of the protocol
- **Simple negotiation**: list of allowed protocols sent to the client, who chooses the one to try first

# XrdSec: flow diagram



## XrdSec: remarks

- Depends on utility module (XrdOuc) only:  
can be easily used in non-XROOTD context
- Working example of standalone client and server programs  
using XrdSec available at

<http://ganis.home.cern.ch/ganis/ROOT/SECURITY/testXrd.tgz>

## Status of plug-ins

- Minimal set wanted:
  - general password-based (*pwd*)
  - GSI certificates-based (*gsi*)
- Multi-tier setups (XROOTD proxy, PROOF?) may require key-based authentication (*key*)
- Additional protocols used in ROOT:
  - Secure Remote Password (*srp*)
  - *ssh* (using *sshd*)
  - *ugid*, uid/gid identification
- Multi-iteration protocols require **tools for parsing buffers**
- Cryptography required

## Password-based plug-in

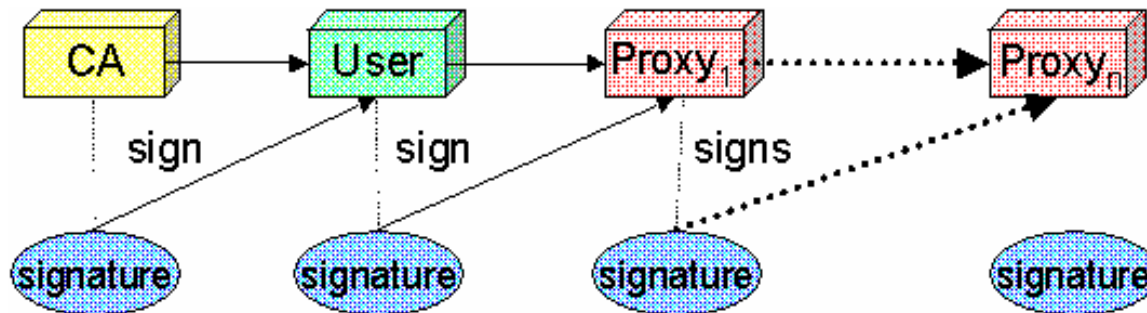
- Derived from existing ROOT code
- Important addition: **centrally administrated credential file**
  - Independence from system usernames and password files
  - Can grant access to users w/o an account on the system
- **Full encryption** with session key and systematic use of **hashing**:
  - password never leaves the client machine
  - information exchanged or stored in central file cannot be used to break in, if compromised
- **Several features as options**:
  - auto-registration, max failures, expiration time, user-defined credential file, use of system password file, auto-login, ...
- **ROOT compatible**: can use **\$HOME/.rootdpass**
- Soon available for testing (end of this week)

# GSI

- Globus Tool Kit infrastructure heavy for pure authentication
- **GridSite** (<http://www.gridsite.org/>) provides “*a toolkit for Grid credentials, GACL access control lists, ...*”
- Approach appealing:
  - Light layer based directly upon OpenSSL
  - **VOMS / GACL compatible**
  - really light: [libgridsite.a](#) is 42 kb.
- Set of C APIs to handle exchanged information (certificates, challenges, ...) fits well into multi-iteration framework

# GSI: delegation

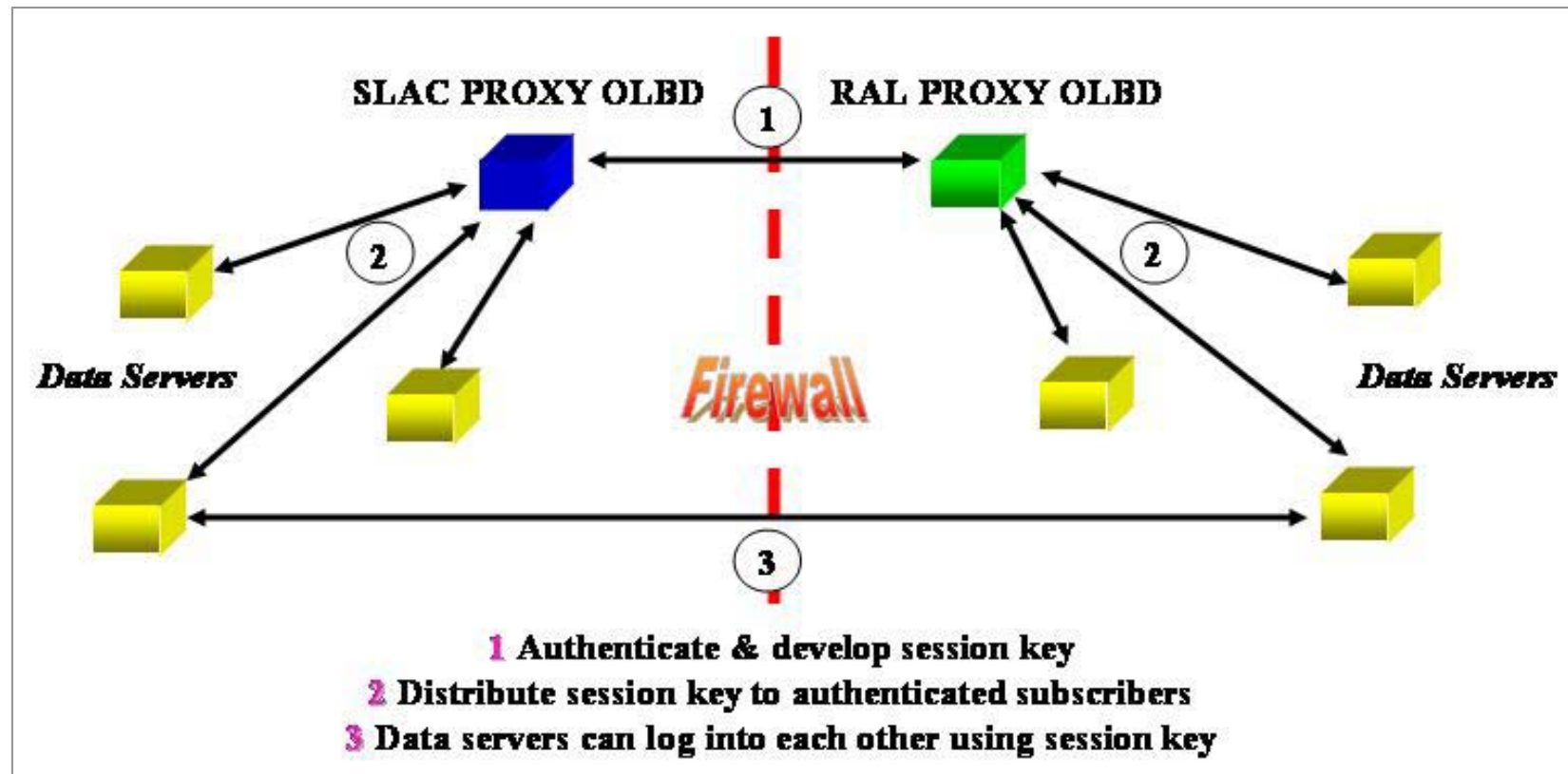
- Used in PROOF in the present approach
- Presently GridSite can check arbitrary certificate chain depths, but does not have the sign functionality needed by



- Proxy protocol specifications is a standard (RFC 3820, July 2004)
- Signing functionality should be a (relatively) minor add-on (under investigation)

## Key-based plug-in

- Used when key previously distributed (XROOTD proxy mode)
- Three-iteration *mutual proof of secret knowledge*



- Keys cached either in memory or in password-like file

## Other protocols used by ROOT

- *SRP* (Secure Remote Password):
  - requires external package from SLAC
  - multi-step mechanism for shared secret setup fits well multi-iteration framework
- *SSH*, fast identification with *UidGid*:
  - require user account on server machine
  - no need of additional setup operations
- Will be provided for ROOT backward compatibility

# Pluggable cryptography

- **Why?**
  - To easy switching between different implementations (OpenSSL, CryptoAPI, Botan, ...)
  - Fits XRD philosophy.
- **How?**
  - Abstract interface for *crypto factory* to get access to relevant crypto functionality:  
*key-agreement, symmetric ciphers, one-way hash, message digest, PKI, handling of X509, ...*
  - *Dynamic* choice: client / server agree on implementation to use and *load / get-handle-of* related factory.
- **Implementations available:**
  - *Ssl*, based on OpenSSL
  - *Local*, limited functionality for *pwd, key* (no ext packages)

## Future plans

- Security Context
  - Presently nothing saved out of successful handshake
  - Will be made available upon request with:
    - an opaque object with secret, ...
    - an expiration time
    - ...
  - Provide protocol methods for encryption / decryption
- Integration in ROOT
  - initially as an alternative
  - require porting on Windows (Unices and MacOS X basically OK)

# Summary

- XrdSec full activation advancing
- Tools for multi-iterations implemented
- Minimal cryptography available
- Password-based plug-in almost ready
- Other plug-ins (starting with GSI) should follow shortly