Enabling Grids for E-science in Europe

Security APIs in LCG-2

Andrea Sciabà
LCG Experiment Integration and Support
CERN IT

Overview

Enabling Grids for E-science in Europe

- Basic security concepts
- Certificates
- Virtual Organisations
- Command line interface
- C/C++ interfaces (GSS-API, GSS Assist)
- Java interface (CoG)
- gSOAP plugins
- Hands-on exercises

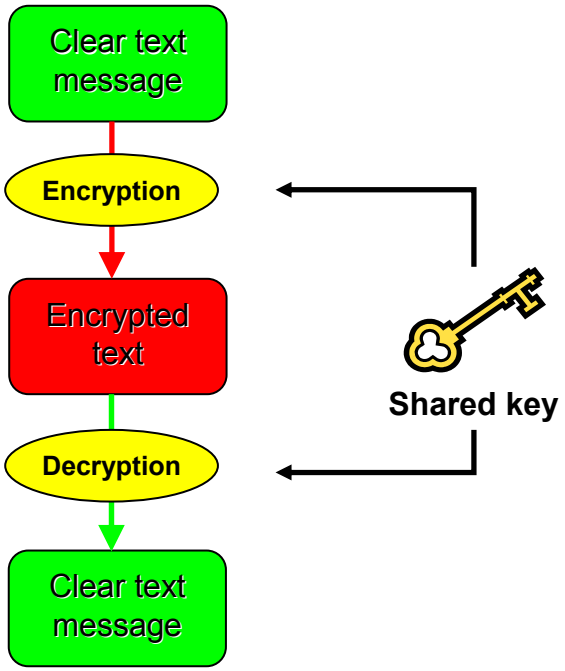
Basic security concepts

Enabling Grids for E-science in Europe

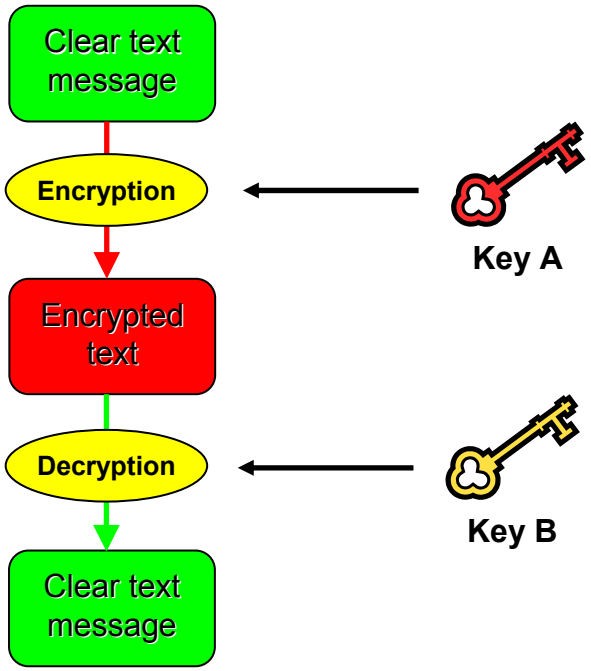
- **Principal**
  - An entity: a user, a program, or a machine
- **Credentials**
  - Some data providing a proof of identity
- **Mechanism**
  - software providing data authentication or confidentiality (e.g. Kerberos, GSI)
- **Authentication**
  - Verify the identity of the peer
- **Authorization**
  - Map an entity to some set of privileges
- **Confidentiality**
  - Encrypt the message so that only the recipient can understand it
- **Integrity**
  - Ensure that the message has not be altered in the transmission
- **Non-repudiation**
  - Impossibility of denying the authenticity of a digital signature

Encryption

- Symmetric encryption: same key (“secret”) used for encryption and decryption
  - Kerberos, DES / 3DES, IDEA

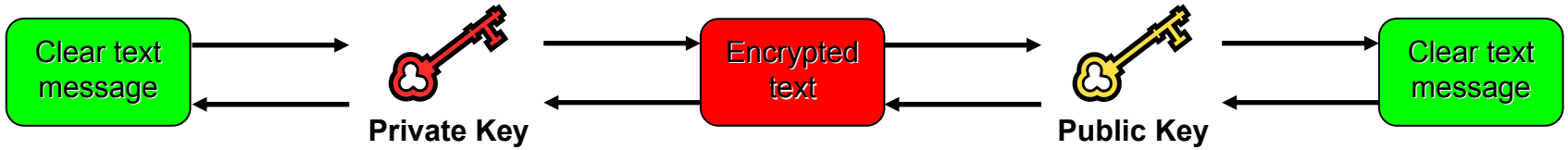


- Asymmetric encryption: different keys used for encryption and decryption
  - RSA, DSA



Public Key Infrastructure

- Provides authentication, integrity, confidentiality, non-repudiation
- Asymmetric encryption

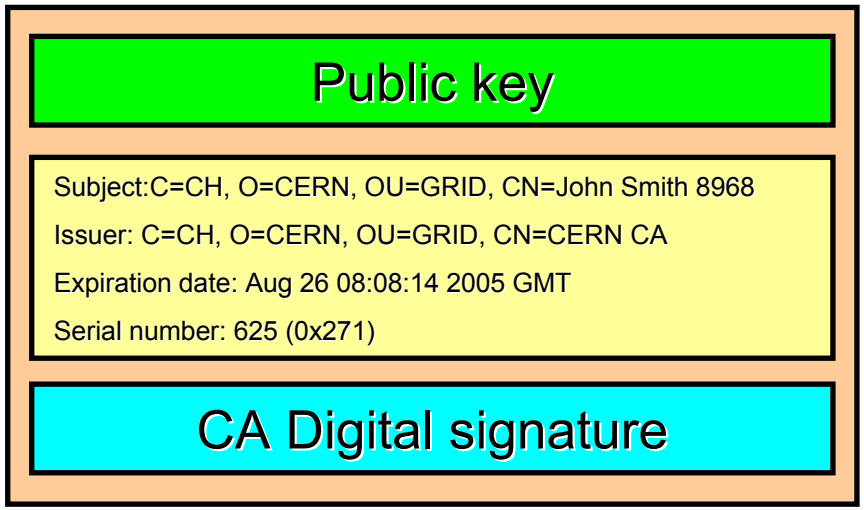


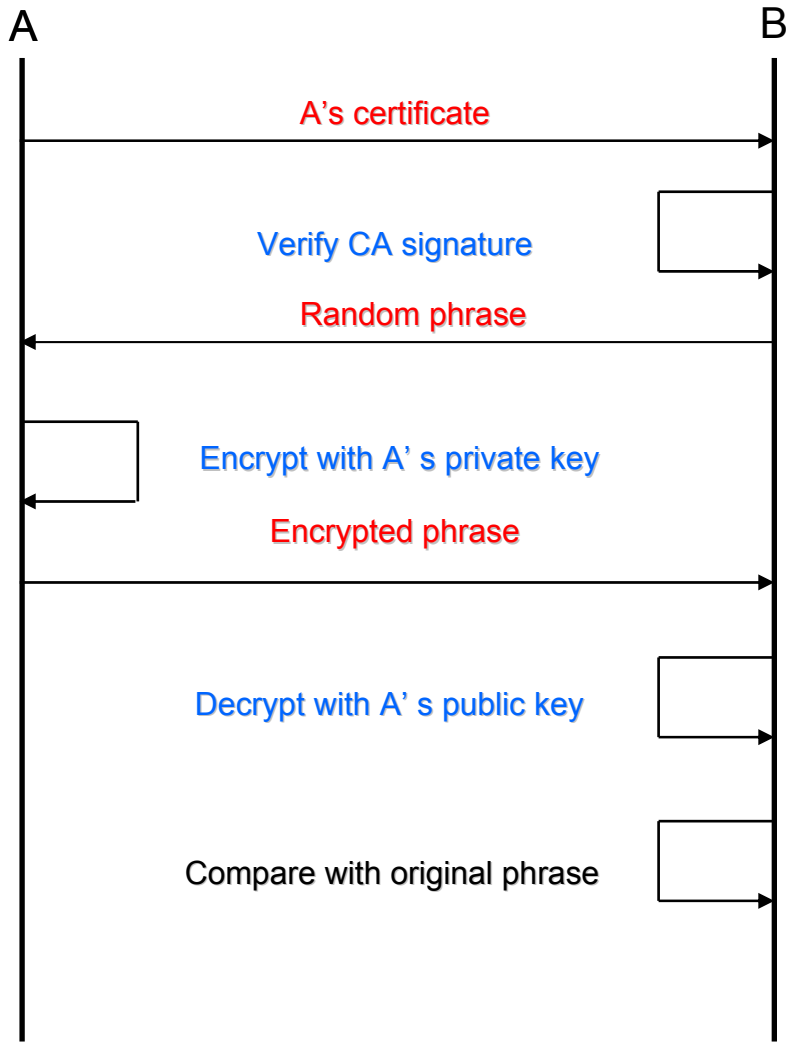
- Digital signatures
  - A hash derived from the message and encrypted with the signer's private key
  - Signature checked decrypting with the signer's public key
- Allows key exchange in an insecure medium using a trust model
  - Keys trusted only if signed by a trusted third party (Certification Authority)
  - A CA certifies that a key belongs to a given principal
- Certificate
  - Public key + information about the principal + CA signature
  - X.509 format most used
- PKI used by SSL, PGP, GSI, WS security, S/MIME, etc.

X.509 certificates and authentication

Enabling Grids for E-science in Europe

Structure of a X.509 certificate





Certification Authorities

Enabling Grids for E-science in Europe

- Issue certificates for users, programs and machines
- Check the identity and the personal data of the requestor
  - Registration Authorities (RAs) do the actual validation
- Manage Certificate Revocation Lists (CRLs)
  - They contain all the revoked certificates yet to expire
- CA certificates are self-signed

- LCG-2 recognizes a given set of CAs
  - https://lcg-registrar.cern.ch/pki_certificates.html

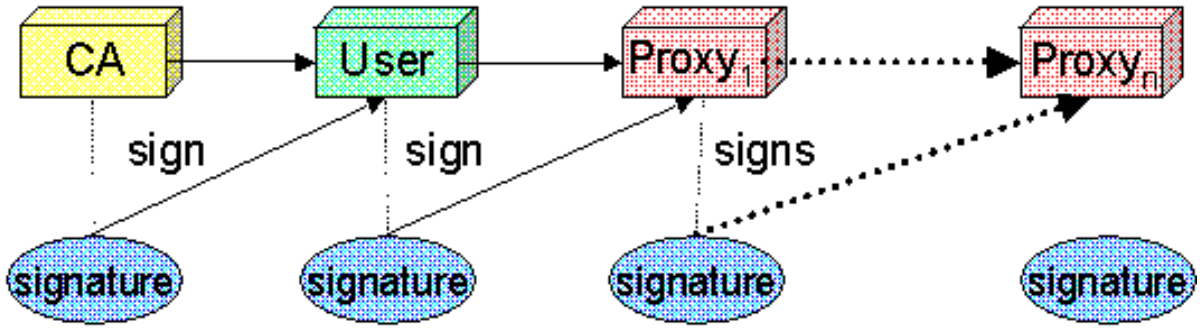
Certificate classification

Enabling Grids for E-science in Europe

- User certificate
  - issued to a physical person
  - DN= `C=CH, O=CERN, OU=GRID, CN =John Smith`
  - the only kind of certificate good for a client, i.e. to send Grid jobs etc.
- Host certificate
  - issued to a machine (i.e. a secure web server, etc.)
  - request signed with a user certificate
  - DN= `C=CH, O=CERN, OU=GRID, CN=host1.cern.ch`
- Grid host certificate
  - issued to a Grid service (i.e. a Resource Broker, a Computing Element, etc.)
  - request signed with a user certificate
  - DN= `C=CH, O=CERN, OU=GRID, CN=host/host1.cern.ch`
- Service certificate
  - issued to a program running on a machine
  - request signed with a user certificate
  - DN= `C=CH, O=CERN, OU=GRID, CN=ldap/host1.cern.ch`

Globus Grid Security Infrastructure (GSI)

- *de facto* standard for Grid middleware
- Based on PKI
- Implements some important features
  - Single sign-on: no need to give one's password every time
  - Delegation: a service can act on behalf of a person
  - Mutual authentication: both sides must authenticate to the other
- Introduces proxy certificates
  - Short-lived certificates including their private key and signed with the user's certificate



More on proxy certificates and delegation

Enabling Grids for E-science in Europe

- Delegation
  - Allowing something else (eg. a file transfer service) to use my credentials
- Proxies can be moved over a network
- Subject identifies the user:
  - User subject: `/C=CH/O=CERN/OU=GRID/CN=Andrea Sciaba 8968`
  - Proxy subject: `/C=CH/O=CERN/OU=GRID/CN=Andrea Sciaba 8968/CN=proxy`
- Full proxy
  - A proxy created from a user certificate or another full proxy with normal delegation
- Limited proxy
  - A proxy created from a proxy with limited delegation, or from another limited proxy
- What does that mean?
  Entities can decide to accept only full proxies. Examples:
  - GridFTP accepts all proxies
  - Globus gatekeeper accepts only full proxies

Virtual Organizations and authorization

Enabling Grids for E-science in Europe

- LCG-2 users MUST belong to a Virtual Organization
  - Sets of users belonging to a collaboration
  - Each VO user has the same access privileges to Grid resources
  - List of supported VOs:
    - https://lcg-registrar.cern.ch/virtual_organization.html
- VOs maintain a list of their members
  - The list is downloaded by Grid machines to map user certificate subjects to local “pool” accounts: only mapped users are authorized in LCG

```
...
"/C=CH/O=CERN/OU=GRID/CN=Simone Campana 7461" .dteam
"/C=CH/O=CERN/OU=GRID/CN=Andrea Sciaba 8968" .cms
"/C=CH/O=CERN/OU=GRID/CN=Patricia Mendez Lorenzo-ALICE" .alice
...
```

grid-mapfile

  - Sites decide which VOs to accept

VOMS, LCAS, LCMAPS

- **Virtual Organization Membership Service**
  - Extends the proxy info with VO membership, group, role and capabilities
- **Local Centre Authorization Service (LCAS)**
  - Checks if the user is authorized (currently using the grid-mapfile)
  - Checks if the user is banned at the site
  - Checks if at that time the site accepts jobs
- **Local Credential Mapping Service (LCMAPS)**
  - Maps grid credentials to local credentials (eg. UNIX uid/gid, AFS tokens, etc.)
  - Currently uses the grid-mapfile (based only on certificate subject)
  - In the near future will map also VOMS group and roles

```
"/VO=cms/GROUP=/cms"                    .cms
"/VO=cms/GROUP=/cms/prod"               .cmsprod
"/VO=cms/GROUP=/cms/prod/ROLE=manager"  .cmsprodman
```

GSI environment variables

- User certificate files:
  - Certificate: X509_USER_CERT (default: `$HOME/.globus/usercert.pem`)
  - Private key: X509_USER_KEY (default: `$HOME/.globus/userkey.pem`)
  - Proxy: X509_USER_PROXY (default: `/tmp/x509up_u<id>`)
- Host certificate files:
  - Certificate: X509_USER_CERT (default: `/etc/grid-security/hostcert.pem`)
  - Private key: X509_USER_KEY (default: `/etc/grid-security/hostkey.pem`)
- Trusted certification authority certificates:
  - X509_CERT_DIR (default: `/etc/grid-security/certificates`)
- Location of the grid-mapfile:
  - GRIDMAP (default: `/etc/grid-security/grid-mapfile`)

Command line interface: certificate and proxy management

Enabling Grids for E-science in Europe

- Get information on a user certificate
  - `grid-cert-info[-help] [-file certfile] [OPTION]...`

`-all`	**whole certificate**
`-subject | -s`	**subject string**
`-issuer | -I`	**Issuer**
`-startdate | -sd`	**Start of validity**
`-enddate | -ed`	**End of validity**

- Create a proxy certificate
  - `grid-proxy-init`
- Destroy a proxy certificate
  - `grid-proxy-destroy`
- Get information on a proxy certificate
  - `grid-proxy-info`

Long term proxy

- Proxy has limited lifetime (default is 12 h)
  - Bad idea to have longer proxy
- However, a grid task might need to use a proxy for a much longer time
  - Grid jobs in HEP Data Challenges on LCG last up to 2 days
- myproxy server:
  - Allows to create and store a long term proxy certificate:
  - `myproxy-init -s <host_name>`
    - `-s <host_name>` specifies the hostname of the myproxy server
  - `myproxy-info`
    - Get information about stored long living proxy
  - `myproxy-get-delegation`
    - Get a new proxy from the MyProxy server
  - `myproxy-destroy`
- A service running continously can renew automatically a proxy created from a long term use proxy and use it to interact with the Grid
  - Examples: automatic job dispatchers or data movers

Security APIs in LCG-2

Enabling Grids for E-science in Europe

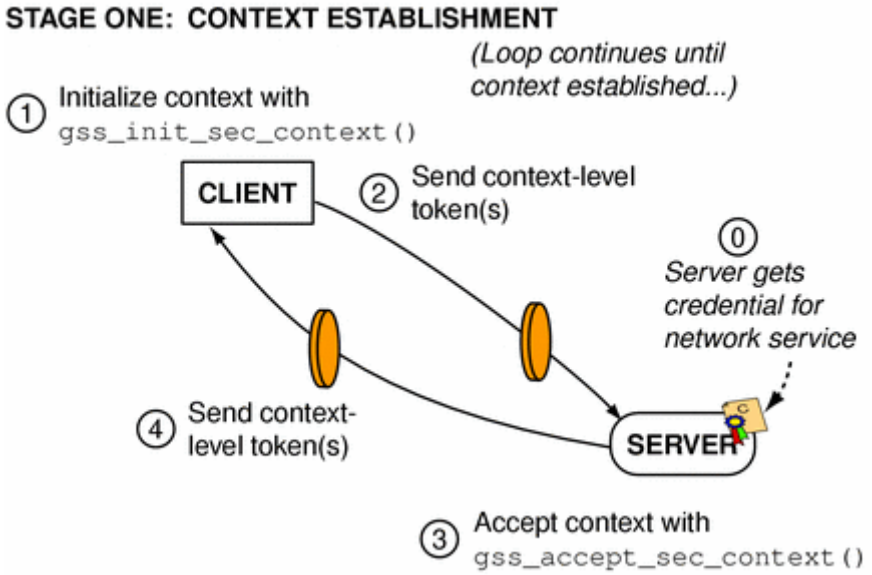
- Currently, there are no security APIs developed specifically by LCG
- The existing APIs come from other projects
  - Authentication
    - Globus GSS-API, GSS Assist, COG Kits (Java and Python)
    - some gSOAP plugins (CERN, Lecce University)
  - Authorization
    - LCAS plugins
    - LCMAPS plugins
    - VOMS API
    - some gSOAP plugins (CERN, Lecce University)
- The documentation is generally not good

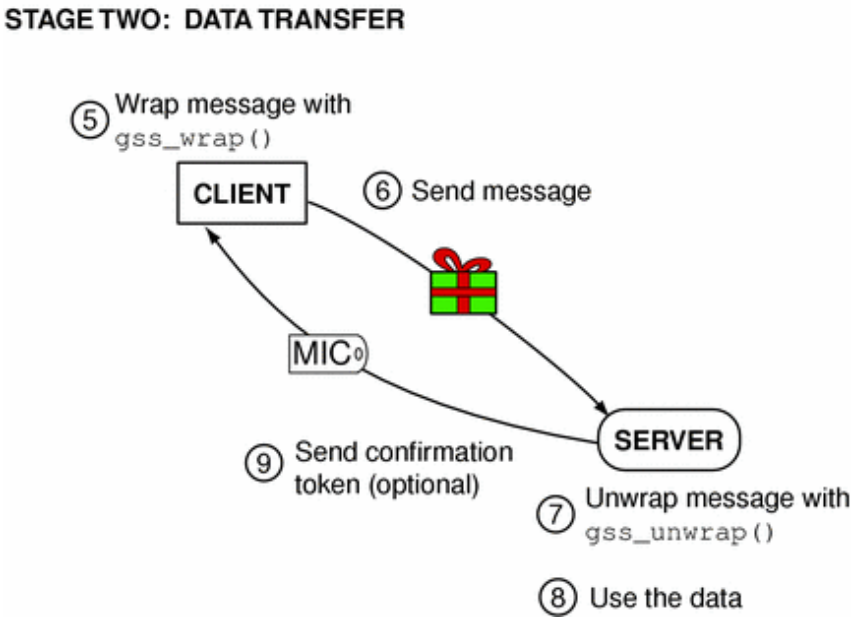
API: GSS-API and GSS Assist

Enabling Grids for E-science in Europe

- GSS-API (Generic Security Services Application Programming Interface) is a generic API for client-server authentication (RFC-2743, 2744)
  - Traditionally, it interfaces to Kerberos
  - The Globus project interfaced it to GSI
  - Communication is kept separate: it just creates data buffers, does not move them
  - Rather complicated to use…
  - **Documentation at http://docs.sun.com/app/docs/doc/816-1331 http://www.gnu.org/software/gss/manual/html_node/index.html**
- GSS-API as user interface to GSI:
  - C API
  - Java API (see later)
- The Globus GSS Assist routines are designed to simplify the use of the GSSAPI: they are a thin layer over them

GSS-API





1. The client initiates a context and prepares a token for the server
2. The token is sent to the server
3. The server interprets the token and prepares a new one to be sent to the client
4. The token is sent to the client
5. Iterate process until authentication process succeeds or fails

5. The client wraps a message for the server
6. The client sends the wrapped message
7. The server unwraps the message
8. The server uses the message
9. The server sends back a confirmation token

GSS-API data types

Enabling Grids for E-science in Europe

- Integers

```
OM_uint32
```

- Strings

```
typedef struct gss_buffer_struct {
   size_t        length;
   void          *value;
} gss_buffer_desc, *gss_buffer_t
```

- Names

```
gss_name_t
```

- OIDs

```
typedef struct gss_OID_desc_struct {
   OM_uint32     length;
   void          *value;
} gss_OID_desc, *gss_OID
```

- OID sets

```
typedef struct gss_OID_set_desc_struct {
   size_t        count;
   gsss_OID      elements;
} gss_OID_set_desc, *gss_OID_set
```

- Credentials

```
gss_cred_id_t
```

- Contexts

```
gss_ctx_id_t
```

More on data types

- Strings (or buffers) are used for character strings and tokens
- Names are an opaque representation of a principal
- Credentials are an opaque representation of a credential
- Object Identifiers (OIDs) are used for
  - Security mechanisms (GSI, Kerberos, etc.)
  - Quality of Protection (QOP) values (the encryption algorithm)
  - Name types
    - GSS_C_NT_HOSTBASED_SERVICE (service@host)
    - GSS_C_NT_USER_NAME (username)
    - Etc.
  - GSS_C_NO_OID: for default mechanism of default QOP (recommended)
- Status codes
  - OM_uint32 major-status: generic GSS-API routine errors
  - OM_uint32 minor-status: mechanism-specific errors
- Secure contexts
  - pairs of data structures (one for each application) with info on their status in terms of security
- Tokens
  - Context level tokens: used for context establishment
  - Per-message tokens: used for data protection (e.g. encrypted messages and cryptographic tags)
  - Inteprocess tokens: used to export and import a secure context between processes

Name manipulation

Enabling Grids for E-science in Europe

- Convert a string to a name and vice versa
  - gss_import_name(), gss_display_name()
- Compare, duplicate names
  - gss_compare_name(), gss_duplicate_name()
- Generate a Mechanism Name (MN), i.e. a mechanism-specific representation of a name
  - gss_canonicalize_name()
- Export a MN in a format suitable for comparison
  - gss_export_name
- Destroy a name
  - gss_release_name()

Object IDentifiers

Enabling Grids for E-science in Europe

- Create an empty OID set
  - gss_create_empty_oid_set()
- Add an OID to a set
  - gss_add_oid_set_member()
- Test if an OID is in a given OID set
  - gss_test_oid_set_member()

Credential management

Enabling Grids for E-science in Europe

- Acquire an existing credential by name
  - gss_acquire_cred()
  - If name is GSS_C_NO_NAME , default credential is used
- Obtain information about a credential
  - gss_inquire_cred(), gss_inquire_cred_by_mech()
  - returns name, lifetime, usage (INITIATE, ACCEPT, BOTH), mechanisms supported
- Destroy a credential handle
  - gss_release_cred()

Context management

Enabling Grids for E-science in Europe

- Establish a secure context
  - gss_init_sec_context()
  - gss_accept_sec_context()
- Retrieve residual duration or other info about context
  - gss_context_time()
  - gss_inquire_context()
- Export/import a context from a process to another by means of an interprocess token
  - gss_export_sec_context(), gss_import_sec_context()
- Destroy a secure context
  - gss_delete_sec_context

Other functions

Enabling Grids for E-science in Europe

- Convert a status code to a textual error message
  - gss_display_status()
- Give available mechanisms
  - gss_indicate_mechs()
- Discard a buffer
  - gss_release_buffer()

Context establishment: client

```
OM_uint32 major_status, minor_status;
gss_ctx_id_t *context;
gss_name_t target_name;
gss_buffer_t input_token, output_token, recv_tok;
input_token = GSS_C_NO_BUFFER;
context = GSS_C_NO_CONTEXT;
do {
    major_status = gss_init_sec_context(&minor_status,
                                        GSS_C_NO_CREDENTIAL,
                                        context,
                                        target_name,
                                        GSS_C_NO_OID,
                                        GSS_MUTUAL_FLAG,
                                        0,
                                        GSS_C_NO_CHANNEL_BINDINGS,
                                        input_token,
                                        NULL,
                                        output_token,
                                        &ret_flags,
                                        NULL);
    if (context == NULL) /* error: exit */
    if (input_token != GSS_C_NO_BUFFER) gss_release_buffer(&minor_status, recv_tok);
    if (major_status != GSS_S_COMPLETE && major_status != GSS_S_CONTINUE_NEEDED) /* error: exit */
    if (output_token->length != 0) {
      /* send output_token to server */
      gss_release_buffer(&minor_status, output_token);
    }
    if (major_status == GSS_S_CONTINUE_NEEDED)
      /* receive recv_tok from server */
    input_token = recv_tok;
} while (major_status == GSS_S_CONTINUE_NEEDED);
```

remote application

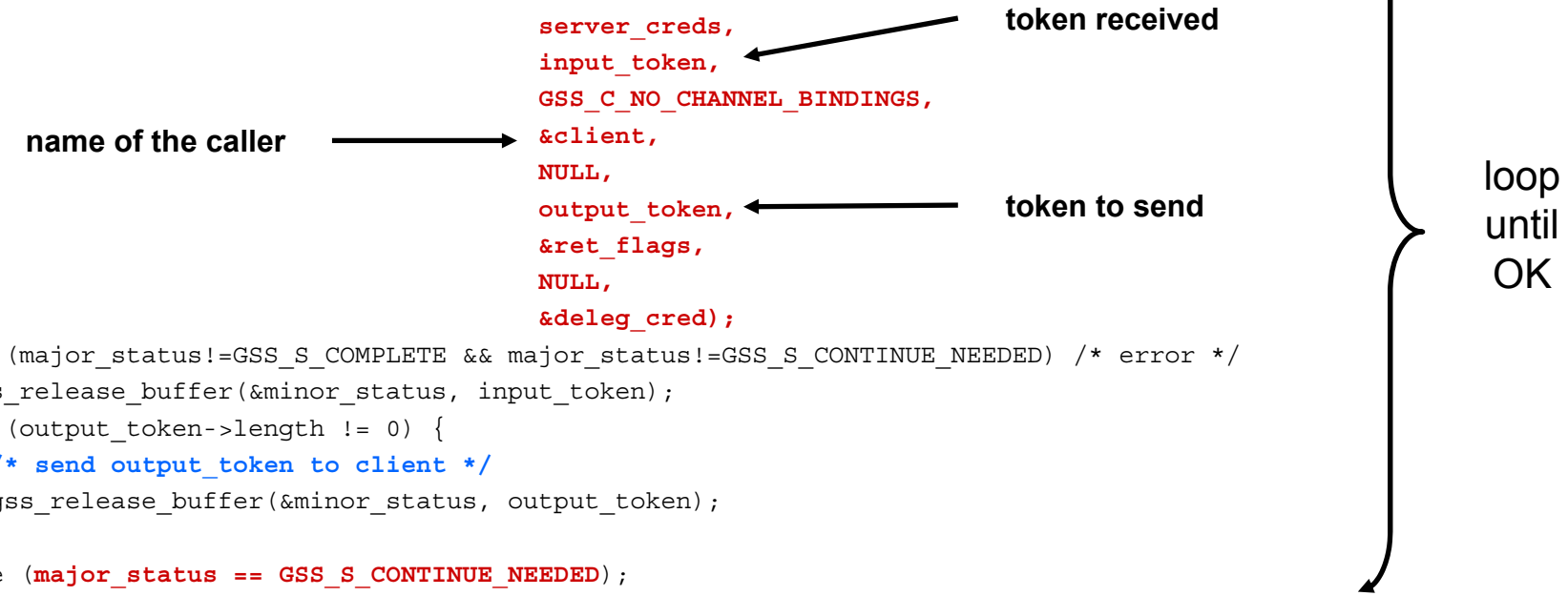
token received

token to send

loop until OK

Context establishment: server

```
OM_uint32 major_status, minor_status, ret_flags;
gss_name_t client;
gss_ctx_id_t *context;
gss_cred_id_t server_creds, deleg_cred;
gss_buffer_t input_token, output_token;
do {
    /* receive input_token from client */
    major_status = gss_accept_sec_context(&minor_status,
                                          context,
                                          server_creds,
                                          input_token,
                                          GSS_C_NO_CHANNEL_BINDINGS,
                                          &client,
                                          NULL,
                                          output_token,
                                          &ret_flags,
                                          NULL,
                                          &deleg_cred);
    if (major_status!=GSS_S_COMPLETE && major_status!=GSS_S_CONTINUE_NEEDED) /* error */
    gss_release_buffer(&minor_status, input_token);
    if (output_token->length != 0) {
      /* send output_token to client */
      gss_release_buffer(&minor_status, output_token);
    }
} while (major_status == GSS_S_CONTINUE_NEEDED);
```



Confidentiality and integrity

Enabling Grids for E-science in Europe

- Generate a cryptographic tag, or message integrity code (MIC) for a message to transfer to the peer application
  - gss_get_mic()
- Verify the received message against the received MIC
  - gss_verify_mic()
- Embed the MIC in the (possibly encrypted) message
  - gss_wrap()
- (possibly decrypt and) verify the embedded MIC
  - gss_unwrap()
- Determine the maximum size for a clear message for a given maximum size of the encrypted message
  - gss_wrap_size_limit()

Example: sending and receiving data

Enabling Grids for E-science in Europe

Encrypting data

```
int conf_req_flag = 1; /* 0 only integrity, <>0 also confidentiality
int conf_state;
gss_buffer_t clear_buffer, wrapped_buffer;
gss_ctx_id_t context;
major_status = gss_wrap(&minor_status, context, conf_req_flag, GSS_C_QOP_DEFAULT,
                        clear_buffer, &conf_state, wrapped_buffer;
```

Decrypting data

```
int conf_state;
gss_buffer_t clear_buffer, wrapped_buffer;
gss_qop_t qop_state;
gss_ctx_id_t context;
major_status = gss_unwrap(&minor_status, context, wrapped_buffer, clear_buffer,
                          &conf_state, &qop_state
```

Globus extensions

- **Credential import and export**
  - To pass credentials from a process to another or storing them in a file
  - Export to 1) an opaque buffer, or 2) a file in GSI native format
  - gss_import_cred(), gss_export_cred()
- **Delegation an any time**
  - A lot more flexible than standard GSS-API delegation
    - Delegation at times other than context establishment
    - Possible to delegate credentials different than those used for context establishment: even for different mechanisms!
      - Ex.: delegate a Kerberos credential over a context established with GSI
  - gss_init_delegation(), gss_accept_delegation()
- **Credentials extension handling**
  - support for credential information other than just the identity
- **Set context options at the server side**
- Documentation
  - http://www.ggf.org/documents/GWD-I-E/GFD-E.024.pdf
  - ${GLOBUS_LOCATION}/include/gcc32dbg/gssapi.h

Example: delegation

```
major_status = gss_init_delegation(&minor_status, context, creds,
                                   GSS_C_NO_OID,          /* default mechanism */
                                   GSS_C_NO_OID_SET,      /* extension OIDs */
                                   GSS_C_NO_BUFFER_SET,   /* extension buffers */
                                   input_token, req_flags, time_req, &time_ret,
                                   output_token);
```

```
major_status = gss_accept_delegation(&minor_status, context,
                                     GSS_C_NO_OID_SET,      /* extension OIDs */
                                     GSS_C_NO_BUFFER_SET,   /* extension buffers */
                                     input_token, req_flags, time_req, &time_ret,
                                     &deleg_creds,
                                     mech_type,
                                     output_token);
```

GSS Assist

- Simpler functions for
  - Credential handle creation

```
major_status = globus_gss_assist_acquire_cred(&minor_status,
                                GSS_C_INITIATE, /* or GSS_C_ACCEPT */
                                &credential_handle);
```

  - Context establishment

```
major_status = globus_gss_assist_init_sec_context(&minor_status,
          credential_handle,
          &context_handle,
          (char *) server_princ,
          GSS_C_DELEG_FLAG|GSS_C_MUTUAL_FLAG,
          &ret_flags,
          &token_status,
          globus_gss_assist_token_get_fd,
          (void *) &socket_fd,
          globus_gss_assist_token_send_fd,
          (void *) &socket_fd);
```

Pointers to functions to send and receive tokens using sockets

  - Display errors, grid-mapfile lookup, etc.
  - Little documentation
    - http://www.globus.org/security/gss_assist.html
    - ${GLOBUS_LOCATION}/include/gcc32/globus_gss_assist.h

Java Commodity Grid (CoG) API

Enabling Grids for E-science in Europe

- Provides a mapping between the Globus Toolkit and Java
- It is implemented in pure Java
- Does not wrap the Globus C implementation
- Allows to interface to all Globus services
- Here we limit ourselves to the GSI library (org.globus.gsi.gssapi)
  - Implements the standard Java GSS-API (org.ietf.jgss)
  - Implements the GSS-API extensions (org.gridforum.jgss)
  - Supports the Globus proxy certificates
- Web site
  - http://www-unix.globus.org/cog/java/
- Manual
  - http://www-unix.globus.org/cog/manual-user.pdf
- API documentation
  - http://www-unix.globus.org/cog/distribution/1.1/api-new/index.html

Java GSS-API

Enabling Grids for E-science in Europe

- Package org.ietf.jgss (RFC 2853)
- Interfaces
  - GSSContext
  - GSSCredential
  - GSSName
- Classes
  - Channel Binding (used to strengthen the authentication)
  - GSSManager (used to instantiate names, credentials and contexts)
  - MessageProp (to specify the level of QOP and confidentiality)
  - Oid
- Exceptions
  - GSSException
- Documentation
  - http://java.sun.com/j2se/1.4.2/docs/api/org/ietf/jgss/package-summary.html

Java GSS-API: how it works

Enabling Grids for E-science in Europe

1. Instantiate a GSSManager (acts as factory)
2. Create principal names and credentials as needed
3. Create a context
4. Engage in a context establishment loop
5. When finished, authentication is complete

- The GSS-API does not perform any communication with the peer; it just generates tokens

Globus GSS-API extensions

Enabling Grids for E-science in Europe

- Package org.gridforum.jgss (RFC 3820)
- Interfaces
  - ExtendedGSSContext
    - Set and get context options, delegate credentials
  - ExtendedGSSCredential
    - Credential export
- Classes
  - ExtendedGSSManager
    - Create a previously imported credential

GlobusGSSManagerImpl

Enabling Grids for E-science in Europe

- Create names
  - GSSName **createName**(String nameStr, Oid nameType)
- Create credentials
  - Default credentials
    - GSSCredential **createCredential**(int usage)
    - GSSCredential **createCredential**(GSSName name, int lifetime, Oid mech, int usage)
  - Exported credentials
    - GSSCredential **createCredential**(byte[] buff, int option, int lifetime, Oid mech, int usage)
- Create contexts
  - Initiator' side
    - GSSContext **createContext**(GSSName peer, Oid mech, GSSCredential cred, int lifetime)
  - Acceptor' side
    - GSSContext **createContext**(GSSCredential cred)

GlobusGSSNameImpl

Enabling Grids for E-science in Europe

- Constructors
  - Default
    - **GlobusGSSName**()
  - From subject
    - **GlobusGSSName**(String name)
  - From subject or service@host name
    - **GlobusGSSName**(String name, Oid nameType)
- Get information
  - boolean **equals**(GSSName another)
  - boolean **isAnonymous**()
  - boolean **isMN**()
  - String **toString**()
    Returns string representation of the name

GlobusGSSCredentialImpl

Enabling Grids for E-science in Europe

- Constructor
  - Anonymous credential
    - **GlobusGSSCredentialImpl**()
  - From GlobusCredential object
    - **GlobusGSSCredentialImpl**(GlobusCredential cred, int usage)
      usage = INITIATE_ONLY, ACCEPT_ONLY, INITIATE_AND_ACCEPT
- Export credentials
  - byte[] **export**(int option)
  - byte[] **export**(int option, Oid mech)
- Get credential information
  - int **getRemainingInitLifetime**(Oid mech)
  - int **getRemainingAcceptLifetime**(Oid mech)
  - int **getRemainingLifetime**()
  - int **getUsage**() int **getUsage** (Oid mech)
  - GSSName **getName**() GSSName **getName**()
  - Oid[] **getMechs**()
  - GlobusCredential **getGlobusCredential**()
  - PrivateKey **getPrivateKey**()
  - X509Certificate[] **getCertificateChain**()
- Destroy
  - void **dispose**()

GlobusGSSContextImpl (1/2)

Enabling Grids for E-science in Europe

- Authenticate
  - byte[] **initSecContext**(byte[] inBuff, int off, int len)
  - int **initSecContext**(InputStream in, OutputStream out)
  - byte[] **acceptSecContext**(byte[] inBuff, int off, int len)
  - void **acceptSecContext**(InputStream in, OutputStream out)
  - boolean **isEstablished**()
- Encrypt, decrypt
  - byte[] **wrap(**byte[] inBuf, int off, int len, MessageProp prop)
  - byte[] **unwrap**(byte[] inBuf, int off, int len, MessageProp prop)
- Create and verify a MIC
  - byte[] **getMIC**(byte[] inBuf, int off, int len, MessageProp prop)
  - void **verifyMIC**(byte[] inTok, int tokOff, int tokLen, byte[] inMsg, int msgOff, int msgLen, MessageProp prop)
- Request context properties
  - void **requestAnonymity**(boolean state)
    void **requestCredDeleg**(boolean state)
    void **requestLifetime**(int lifetime)
    void **requestReplayDet**(boolean state)
    void **requestConf**(boolean state)
    void **requestInteg**(boolean state)
    void **requestMutualAuth**(boolean state)
    void **requestSequenceDet**(boolean state)

GlobusGSSContextImpl (2/2)

Enabling Grids for E-science in Europe

- Get context properties
  - boolean **getAnonymityState**()
  - boolean **getCredDelegState**()
  - boolean **getMutualAuthState**()
  - boolean **getSequenceDetState**()
  - int **getLifetime**()
  - GSSName **getSrcNname**()
  - Oid **getMech**()
  - boolean **getConfState**()
  - boolean **getIntegState**()
  - boolean **getReplayDetState**()
  - boolean **isProtReady**()
  - boolean **isInitiator**()
  - GSSName **getTargName**()
- Delegation
  - byte[] **initDelegation**(GSSCredential credential, Oid mechanism, int lifetime, byte[] buf, int off, int len)
  - byte[] **acceptDelegation**(int lifetime, byte[] buf, int off, int len)
  - boolean **isDelegationFinished**()
  - GSSCredential **getDelegatedCredential**()
  - GSSCredential getDelegCred()
- Set, get context options
  - void **setOption**(Oid option, Object value)
  - Object **getOption**(Oid option)
- Destroy
  - void dispose()

Example: context establishment: client

```
Socket s = new(host, port);
OutputStream out = s.getOutputStream();
InputStream in = s.getInputStream();
byte[] inToken = new byte[0];
byte[] outToken = null;
GSSManager manager = new GlobusGSSManagerImpl();
GSSName targetName = null;
targetName = manager.createName("host@" + host, GSSName.NT_HOSTBASED_SERVICE);
// Establish secure context
ExtendedGSSContext context = (ExtendedGSSContext)manager.createContext(targetName,
                              GSSConstants.MECH_OID, null, lifetime);
context.requestConf(true);
context.setOption(GSSConstants.DELEGATION_TYPE, GSIConstants.DELEGATION_TYPE_FULL);
while (!context.isEstablished()) {
  outToken = context.initSecContext(inToken, 0, inToken.length);
  if (outToken != null) {
    out.write(outToken);
    out.flush();
  }
  if (!context.isEstablished()) {
    inToken = SSLUtil.readSslMessage(in);
  }
}
// Send encrypted message
String msg = "Hello world\n";
btye[] tmp = msg.getBytes();
outToken = context.wrap(tmp, 0, tmp.length, null);
out.write(outToken);
out.flush();
```

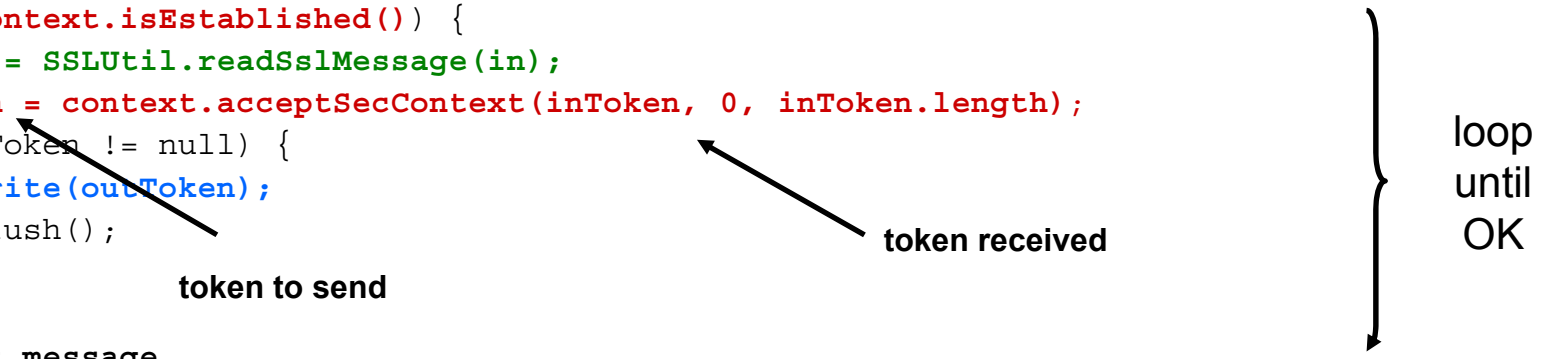
token to send

token received

loop until OK

Example: context establishment: server

```
Socket s;
OutputStream out = s.getOutputStream();
InputStream in = s.getInputStream();
byte[] inToken = null;
byte[] outToken = null;
GSSManager manager = new GlobusGSSManagerImpl();
// Establish secure context
ExtendedGSSContext context =
    (ExtendedGSSContext)manager.createContext((GSSCredential)null);
context.requestConf(true);
context.setOption(GSSConstants.REJECT_LIMITED_PROXY, true);
while (!context.isEstablished()) {
  inToken = SSLUtil.readSslMessage(in);
  outToken = context.acceptSecContext(inToken, 0, inToken.length);
  if (outToken != null) {
    out.write(outToken);
    out.flush();
  }
}
// Decrypt message
inToken = SSLUtil.readSslMessage(in);
outToken = context.unwrap(inToken, 0, inToken.length, null);
System.out.println(new String(outToken));
```



Example: credential delegation

Enabling Grids for E-science in Europe

Client

```
inToken = new byte[0];
while(!context.isDelegationFinished()) {
    outToken
            = context.initDelegation(null, null, 10000, inToken, 0, inToken.length);

    if (outToken != null) {
            out.write(outToken);
            out.flush();
    }

    if (!context.isDelegationFinished()) {
            inToken = SSLUtil.readSslMessage(in);
    }
}
```

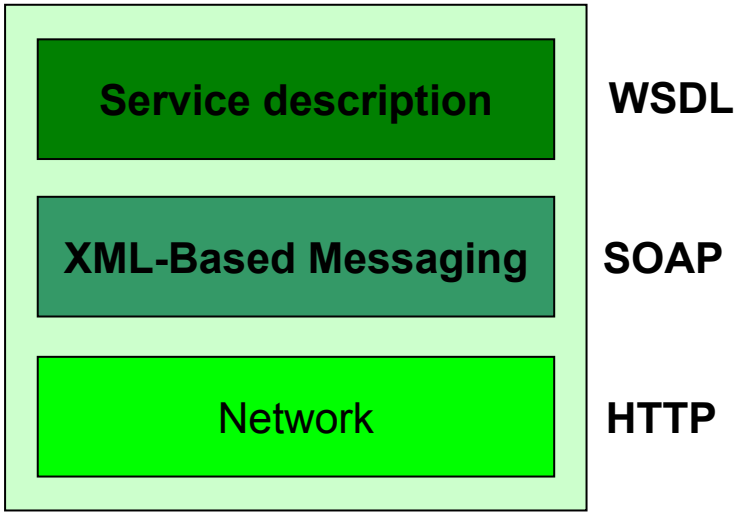
Server

```
while (!context.isDelegationFinished()) {
    inToken = SSLUtil.readSslMessage(in);
    outToken = context.acceptDelegation(10000, inToken, 0, inToken.length);

    if (outToken != null) {
            out.write(outToken);
            out.flush();
    }
}
```

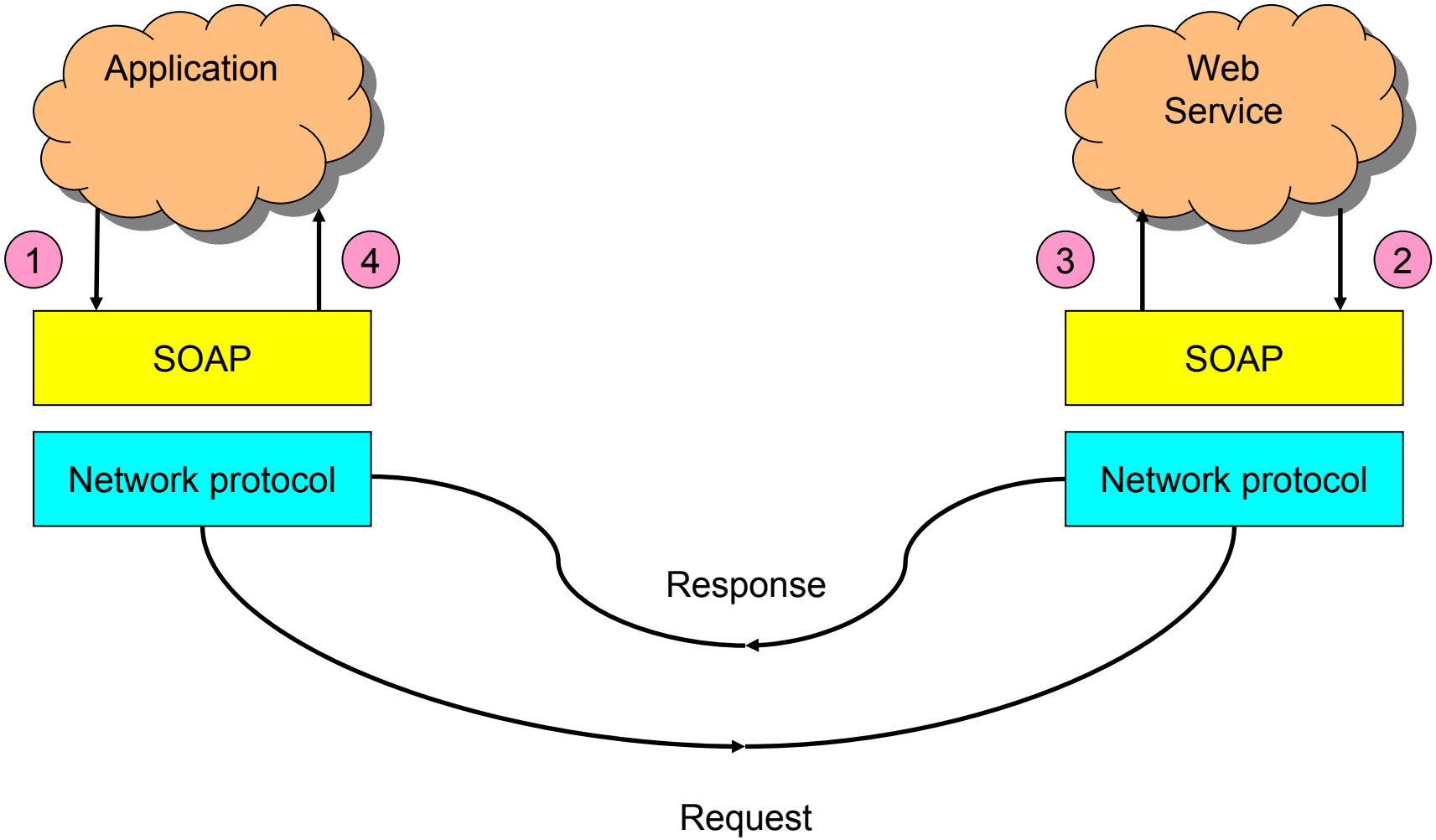
Web services

- What is a Web Service?
  - a software service exposed on the Web through SOAP and described with a WDSL
- What is SOAP (Simple Object Access Protocol)?
  - an XML-based, lightweight protocol used for exchange of information and for Remote Procedure Calls
  - uses HTTP for transport
- What is WSDL (Web Services Description Language)?
  - an XML document describing a set of SOAP messages and how they are exchanged
  - describes the location of the Web Service
  - the WSDL file defines everything required to write a program to work with an XML Web service



Web Services Stack

Web services



Web Service Security

- Message level security
  - WS-Security
    - set of SOAP extensions to implement integrity and confidentiality in Web Services
    - <Security> header contains the security-related information
    - http://www-128.ibm.com/developerworks/library/ws-secure/
  - WS-SecureConversation
    - defines how to establish secure contexts and exchange keys
  - Used in Globus Toolkit 3
- Transport level security
  - SOAP messages are transmitted encrypted
  - used by some gSOAP GSI plugins
- Only transport level security covered in this tutorial

gSOAP

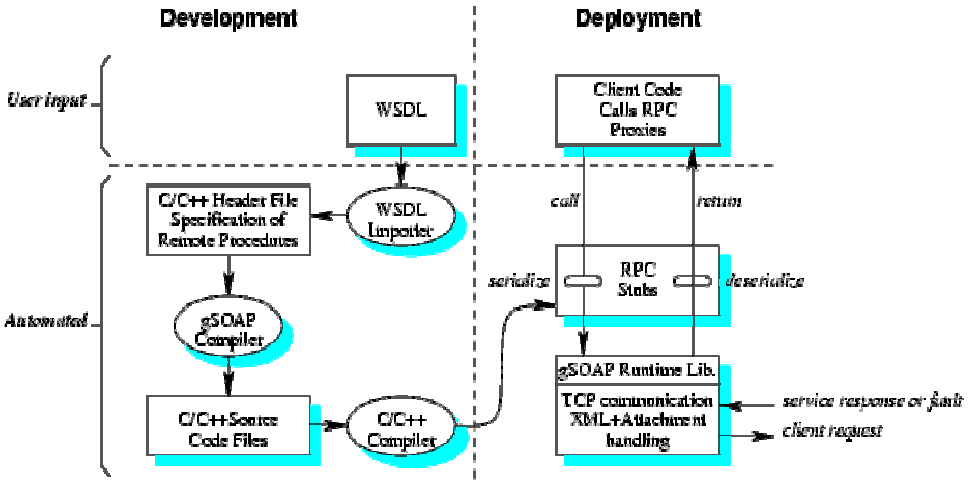
Enabling Grids for E-science in Europe

- gSOAP is a toolkit that offers an XML to C/C++ language binding to make easy the development of SOAP/XML Web services in C and C/C++
- Allows to generate code to write a client to interact with a Web Service starting from the service description published by the WS itself (WSDL file)
- Allows to generate code to implement a WS in C/C++ starting from the WS description
- Generates the code which provides a transport layer with HTTP on top of TCP/IP and to serialize/deserialize structured data

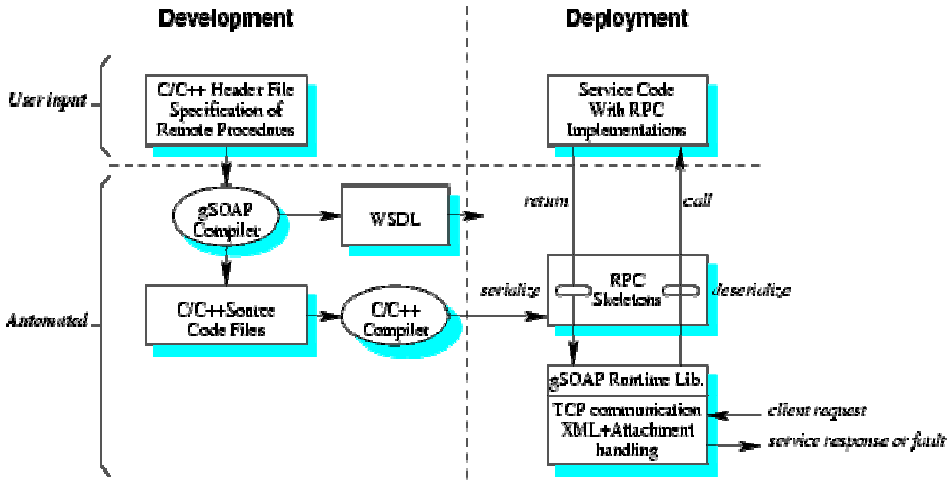
- For more information, look at http://www.cs.fsu.edu/~engelen/soap.html

gSOAP: how to use it

1. Generate the header file from the WSDL description
   - `wsdl2h -o [-c] outfile.h infile.wsdl`
2. Generate the stubs (for clients) and the skeletons (for services)
   - `soapcpp2 [-c] [file]`
   - `soapClient.cpp` stub routine
   - `soapServer.cpp` skeleton routine
   - `soapC.cpp` serializer and deserializer
   - `soapH.h` include file for user code
   - `soapStub.h` include file for stub data types and methods
   - `Service.nsmap` namespace aliases
   - `soapServiceProxy.h` C++ proxy class



Client development



Web Service development

Example: client development

- Generate the header file with the command

```
$ wsdl2h -o quote.h http://websrv.cs.fsu.edu/~engelen/calc.wsdl
```

```
//gsoap ns service name:        calc
//gsoap ns service style:       rpc
//gsoap ns service encoding:    encoded
//gsoap ns service namespace:   http://websrv.cs.fsu.edu/~engelen/calc.wsdl
//gsoap ns service location:    http://websrv.cs.fsu.edu/~engelen/calc.cgi

//gsoap ns schema namespace: urn:calc
int ns__add(double a, double b, double *result);
```

- Compile the header file with the command

```
$ soapcpp2 calc.h
```

- Write the client

```
#include "soapH.h"
#include "calc.nsmap"

const char server[] = "http://websrv.cs.fsu.edu/~engelen/calcserver.cgi";

int main(int argc, char **argv)
{ struct soap soap;
  double a, b, result;
  soap_init(&soap);
  a = strtod(argv[1], NULL);
  b = strtod(argv[2], NULL);
  soap_call_ns__add(&soap, server, "", a, b, &result);
 if (soap.error)
    soap_print_fault(&soap, stderr);
  else
    printf("result = %g\n", result);
  return 0;
}
```

Example: Web Service development

Enabling Grids for E-science in Europe

- Generate the header file with the command

```
$ wsdl2h -o quote.h http://websrv.cs.fsu.edu/~engelen/calc.wsdl
```

- Compile the header file with the command

```
$ soapcpp2 calc.h
```

- Write the client

```
#include "soapH.h"
#include "calc.nsmap"

int main(int argc, char **argv) {
  int m, s; /* master and slave sockets */
  struct soap soap;
  soap_init(&soap);
  m = soap_bind(&soap, NULL, atoi(argv[1]), 100);
  fprintf(stderr, "Socket connection successful: master socket = %d\n", m);
  for ( ; ; ) {
    s = soap_accept(&soap);
    fprintf(stderr, "Socket connection successful: slave socket = %d\n", s);
    if (s < 0) {
      soap_print_fault(&soap, stderr);
      exit(-1);
    }
    soap_serve(&soap);
    soap_end(&soap);
    }
  }
  return 0;
}

int ns__add(struct soap *soap, double a, double b, double *result)
{ *result = a + b;
  return SOAP_OK;
}
```

gSOAP plug-ins

- What is a gSOAP plug-in?
  - a way to extend the gSOAP capabilities
  - adds custom data to the gSOAP run-time
  - overrides standard gSOAP function callbacks by custom callbacks
- Relevant functions:
  - `int soap_register_plugin_arg()`
    - to initialize the plug-in and associate its data to the gSOAP run-time
  - `void* soap_lookup_plugin()`
    - to access the plug-in internal data

GSI plug-in #1: CGSI_gSOAP

- Developed by the CERN CASTOR team
- Used by some data management LCG tools and APIs
- Code: http://savannah.cern.ch/files/index.php?group=castor&thread_max=1&highlight=
- API docs: http://bcouturi.home.cern.ch/bcouturi/cgsi_gsoap/
- Functions:
  - Create gsoap object and register plug-in at the same time
    - `int soap_cgsi_init (struct soap *soap, int cgsi_options)`
  - Constructors
    - `int cgsi_plugin (struct soap *soap, struct soap_plugin *plugin, void *arg)`
    - `int client_cgsi_plugin (struct soap *soap, struct soap_plugin *plugin, void *arg)`
    - `int server_cgsi_plugin (struct soap *soap, struct soap_plugin *plugin, void *arg)`
  - Get information
    - `int is_context_established (struct soap *soap)`
    - `int get_client_dn (struct soap *soap, char *dn, size_t dnlen)`
    - `int get_client_username (struct soap *soap, char *username, size_t dnlen)`
  - Credential management
    - `int export_delegated_credentials (struct soap *soap, char *filename)`
    - `int has_delegated_credentials (struct soap *soap)`
    - `int set_default_proxy_file (struct soap *soap, char *filename)`
    - `void clear_default_proxy_file (int unlink_file)`

GSI plug-in #2: GSI Plugin for gSOAP

Enabling Grids for E-science in Europe

- Developed by M. Cafaro, D. Lezzi (Lecce Univ.) and R. Van Engelen (Florida State Univ.)
- Not used in LCG, but used elsewhere
- Only trasport-level security, but also plans to support message-level security
- Web page: http://sara.unile.it/~cafaro/gsi-plugin.html
- Documentation: README in the tarball distribution
- Client-side functions:
  - `int gsi_connect()`: connecting to a server
  - `int gsi_set_delegation()`: request delegation
  - `int gsi_set_replay()`: request replay attack detection
  - `int gsi_set_sequence()`: request out-of-sequence message detection
  - `int gsi_set_confidentiality()`: request encryption
  - `int gsi_set_integrity()`: request integrity
- Server-side functions
  - `int gsi_listen()`: listen for incoming connections
  - `int gsi_accept()`: accept incoming connections
- General functions
  - `int globus_gsi()`: plugin constructor
  - `int gsi_acquire_credential()`: acquire credentials
  - `int gsi_authorization_callback()`: callback to define a custom authorization policy

Plug-in comparison

Enabling Grids for E-science in Europe

Feature	CGSI	GSI Plug-in
based on GSS-API	√	√
debugging framework	√	√
GSS error reporting	√	√
support for IPv6		√
client development	√	√
server development	√	√
muthual authentication	√	√
authorization	√	√
credential delegation	√	√
connection caching	√	√

The two plug-ins are interoperable

Conclusions

- A lot of security APIs to deal with in LCG
- A lot of different options
  - C/C++
  - Java
  - C/C++ Web Services
- Probably a lot more things around, but not in LCG
  - Python (not covered here)
  - Java Web Services (not covered here)
  - obviously, everything can be used if it is part of the application software, i.e. not provided by LCG itself
- “Hands-on” session:
  - play with simple client/server applications to exercise the different APIs

Enabling Grids for E-science in Europe

LCG Security API

“Hands-on” session

List of exercises

Enabling Grids for E-science in Europe

1. Client/server application with GSS-API in C
2. Client/server application with GSS-API in Java
3. Client/server application with gSOAP and the CGSI plug-in
4. Client/server application with gSOAP and the GSI plug-in

You can freely choose which ones to try

You WILL need to look at the API documentation linked in these slides

Comments in the code mark the places where code must be added for the exercises

GSS-API in C

- Get the code
  - `$ cp -a /home/sciaba/gss/gssapi ~/gssapi`
- Compile the code
  - `$ cd ~/gssapi`
  - `$ make`
- Run the code
  - create a proxy with `grid-proxy-init`
  - from a session, run `./listen`
  - from another session, run
    `$ ./connect localhost <port>`
  - where `<port>` is the one returned by `./listen`

C GSS-API exercises

Enabling Grids for E-science in Europe

- Exercise 1
  - let the server print if confidentiality, integrity and mutual authentication are enabled for the context
- Exercise 2
  - enable confidentiality, integrity and mutual authentication if they are not
- Exercise 3 (optional)
  - look at the code in gssapi_delegation_test
  - export the delegated credentials to a file
  - print the buffer returned by gss_export_cred() (it contains the path of the delegated proxy)
  - inspect the proxy with grid-proxy-info

GSS-API in Java

- Get the code
  - `$ cp -a /home/sciaba/gss/cog ~/cog`
- Compile
  - `$ cd ~/cog`
  - `./compile`
- Run the code
  - create a proxy with `grid-proxy-init`
  - from a session, source the enviroment with `. env.sh and` run
    `$ java GssServer`
  - from another session, source the enviroment with `. env.sh` and run
    `$ java GssClient localhost <port>`
  - where `<port>` is the one returned by `GssServer`

Java GSS-API exercises

Enabling Grids for E-science in Europe

- Exercise 1
  - look at the online help of GssServer and GssClient (`-help` option)
  - using the options, establish a context with/without confidentiality and anonymity (hint: also the server must be invoked with the proper option)
- Exercise 2
  - using the options:
  - use standard credential delegation (requested at context initiation)
  - use Globus extended delegation (hint: `-deleg-type` extended for both client and server)

CGSI plug-in

- Get the code
  - `$ cp -a /home/sciaba/gsoap/example-cgsi ~/example-cgsi`
- Compile the code
  - `$ cd ~/example-cgsi`
  - `$ make`
- Create a “grid-mapfile”
  - `$ SUBJECT=`grid-cert-info -subject`
  - `$ echo \"$SUBJECT\" user > grid-mapfile`
- Launch the server
  - `$ export GRIDMAP=grid-mapfile`
  - `$ ./calcserver -p <port>`
    (to avoid clashes use 10000+account no.)
- Launch the client from another session
  - `$ ./calcclient -p <port> add 10 5`
- Debug information
  - `CGSI_TRACE=1`
  - `CGSI_TRACEFILE=<file>`
- If you need it, `make distclean` with remove the binaries

CGSI plug-in exercises

Enabling Grids for E-science in Europe

- Exercise 1
  - modify the server code to check if the client has delegated credentials, and if so, export them to a file
  - make the client to delegate its credentials to the server

gSOAP GSI plug-in

Enabling Grids for E-science in Europe

- Get the code
  - `$ cp -a /home/sciaba/gsoap/example-gsi-plugin \ ~/example-gsi-plugin`
- Compile the code
  - `$ cd ~/example-gsi-plugin`
  - `$ make`
- Create an authorization file
  - `$ SUBJECT=`grid-cert-info -subject``
  - `$ echo "$SUBJECT" > authorized_dn`
- Launch the server
  - `$ ./calcserver -p <port>`

  (to avoid clashes use 10000+account no.)
- Launch the client from another session
  - `$ ./calcclient -p <port> add 10 5`
- Debug information
  - `export GSI_PLUGIN_DEBUG_LEVEL={0, 1, 2, 3, 4}`

gSOAP GSI plug-in exercises

Enabling Grids for E-science in Europe

- Exercise 1
  - make the client delegate the credentials to the server
  - make the server export the delegated credentials (if any) to a file
  - Hints:
    - refer to `/home/sciaba/gsoap/gsoap-gsi-plugin-2.4.2/include/gsi.h`
    - use gss_export_cred() from the extended GSS-API