

The Globus Toolkit: A Primer for Site Administrators

Von Welch
welch@mcs.anl.gov

Abstract

This document is intended to give an overview of the Globus Toolkit™ to system administrators and policy makers at sites interested in installing the Globus Toolkit. In particular it discusses the security aspects of the Globus system and how local site administrators can configure the security components of Globus to meet their local site policy.

1 Introduction

The Globus Toolkit™ includes software for security, information infrastructure, resource management, data management, and communication in order to enable Grid computing. The Grid [1] refers to an infrastructure that enables the integrated, collaborative use of high-end computers, networks, databases, and scientific instruments owned and managed by multiple organizations. Grid applications often involve large amounts of data and/or computing and often require secure resource sharing across organizational boundaries, and are thus not easily handled by today's Internet and Web infrastructures.

Over the past five years the Globus Toolkit(tm) has gained wide acceptance in research communities around the world, with production deployments such as the NASA Information Power Grid [2], the National Computational Science Alliance (NCSA) [3], the National Partnership for Advanced Computational Infrastructure (NPACI) [4], the DOE Science Grid [5], and the EU DataGrid.

The Globus Toolkit is now also receiving commercial support. At SuperComputing 2001, a dozen vendors: Compaq, Cray, Entropia, Fujitsu, Hitachi, IBM, Microsoft, NEC, Platform, SGI, Sun and Veridian, announced porting of the Globus Toolkit to their platform or products, or are giving financial support to the Globus Project.

The Globus Toolkit attributes no small portion of its success to the fact that it was designed with security as a key component. In particular the Globus Project has followed the principal of using standard, widely-accepted community security standards and tools such as X.509 Certificates [6], SSL/TLS [7] and the OpenSSL library [8].

The Toolkit also follow the tenet of layering on top of existing site infrastructure and policy. It does not try to replace a site's current infrastructure but instead provides a

flexible layer on top of it to provide an interface for Grid computing. The toolkit is also configurable and flexible to allow it to conform to a site's existing policy.

This paper is meant to give system administrators and policy makers a simple overview of the Globus Toolkit(tm) (Version 2.0 released in April of 2002) and the Grid Security Infrastructure (GSI), on which the Globus Toolkit bases its security services. It explains how the administrator can control access without having to understand all the details.

In Section 2 we give an overview of the Grid Security Infrastructure and describe a number of its features in detail. In Section 3 we describe the most commonly deployed Globus Toolkit components. In Section 4 we discuss how the Globus Toolkit interacts with firewalls. In Section 5 we discuss using Globus with Kerberos. In Section 6 we answer some common questions about the Globus Toolkit and the Globus Project. In Section 7 we summarize this information and provide pointers to sources of more detail.

2 Overview of the Grid Security Infrastructure (GSI)

The Grid Security Infrastructure (GSI) provides the security services for the Globus Toolkit. It is based on standard technologies, namely Public Key Infrastructure (PKI), X.509 Certificates and Secure Socket Layer (SSL). These standard technologies were chosen due to their popular use by the general computing community and hence are well tested and understood. It is assumed the reader is familiar with these standard technologies, if not descriptions of these technologies are available in Appendix 2.

One concept that is critical to understanding GSI is the difference between *authentication* and *authorization*. These concepts are often lumped together but are distinct in GSI. *Authentication* is the act of one party proving its identity to another. Authentication by itself does not grant the a part any rights, but is usually a step to determining those rights. *Authorization* is the process of establishing what rights, if any, that identity has on a resource. ‘

In this section we give a overview of GSI on both the client and server side, then we go into depth on GSI specifics: long-term credentials, proxy credentials, establishing a GSI-secured channel, and delegation.

2.1 GSI Client Side

Figure 1 shows the operation of GSI with a typical client. Shown are the various credentials (long-term and proxy), tools (grid-cert-request and grid-proxy-init, both part of the Globus Toolkit) and libraries (GSI), that a user would use when using GSI. The process is broken down into four steps: (1) getting long-term credentials, (2) creating proxy credentials from the long-term credentials, and (3) using the GSI libraries and the proxy credentials to authenticate and securely communicate with a server and (4) optional delegation. In this section we explain each of these steps in detail.

These steps are discussed in detail in the later subsections of this section.

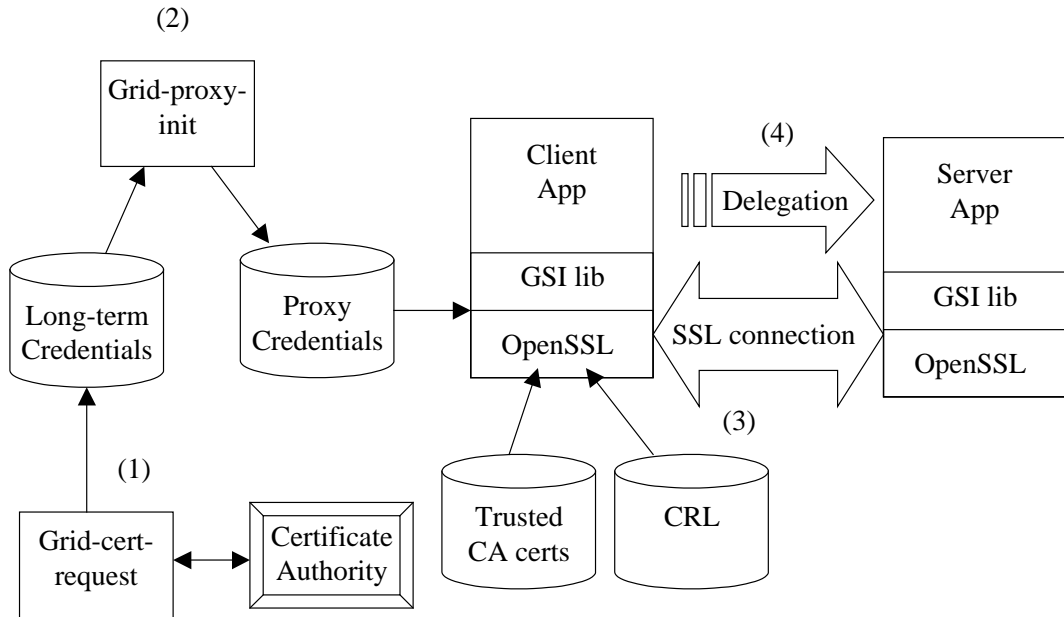


Figure 1: GSI Client Side

2.2 GSI Server Side

Figure 2 shows the operation of GSI in a typical server (e.g. GridFTP server, Gatekeeper, MDS server). Shown are the server's credentials and libraries that the server application uses as well as the tools (grid-cert-request, part of the Globus Toolkit), and configuration files used by the administrator to create policy. Five steps are shown, (1) the administrator getting long-term credentials for the server, (2) a client connecting with a GSI-enabled client, (3) the server's GSI libraries checking the CA that signed the user's X.509 certificate to make sure it is trusted by local policy, (4) optionally checking to see if the user's certificate has been revoked (invalidated) by the issuing CA, and (5) the server's GSI libraries checking the grid-mapfile configuration file to make sure the user is allowed to use the resource by local policy and what local identity (e.g. Unix userid) they should run under. In this section we explain each of these steps in detail.

Note that steps 1-4 are the GSI authentication process and step 5 is the GSI authorization process. All of these steps are discussed in detail in the following subsections of this section.

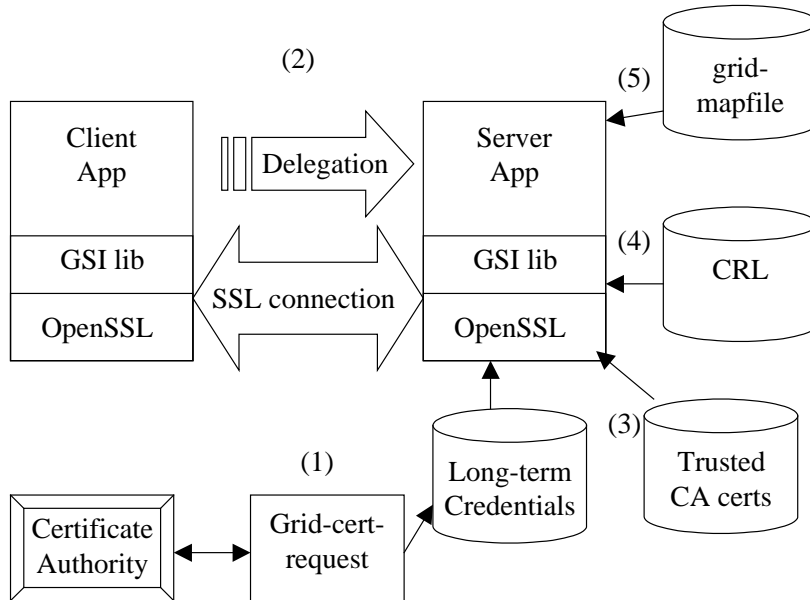


Figure 2: GSI Server side

2.3 Long-term credentials

Step 1 of both Figure 1 and Figure 2, shows the use of the grid-cert-request tool to obtain a set of long-term credentials to identify the entity (user or server) in question.

2.3.1 GSI Credential Format

GSI uses standard X.509 identity credentials. As shown in Figure 3, these credentials are composed of a private key, generated by the user, and an X.509 certificate issued by a Certificate Authority (CA). The certificate is basically a cryptographically signed binding between the private key and an identity issued by the CA.

Identity is established by presenting the certificate and then proving possession of the private key. Since the entity owning the identity credentials has sole possession of the private key, this proves the entity can assert the identity in the certificate. If a party presented with this proof also trusts the CA that issued the certificate this is acceptable proof of the user's identity.

Note that the terms "certificate" and "credentials" are often used interchangeably. In this document we use the term "certificate" to refer to the public portion of the credentials and the term "credentials" to refer to the collection of all items needed to assert and prove possession of an identity – the private key and (one or more) certificate.

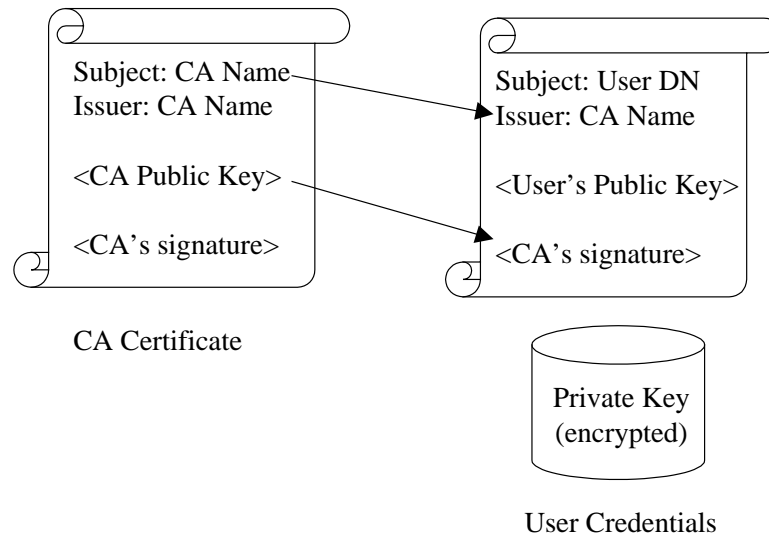


Figure 3: Long-term GSI Credentials

2.3.2 Credential Identity

An GSI identity is the distinguished name (DN) in the entity's X.509 certificate. In a user certificate the identity can be an arbitrary string as decided by the CA. The only requirement made by GSI is that it is globally and temporally unique. To obtain this it typically contains information such as the user's organization and real name, but Globus and GSI treat this as an opaque string. User's identities are used in the Globus grid-mapfile authorization file described later in Section 2.6.2.

A server identity is used by GSI clients to verify they are connecting to the correct server. This identity must also be unique, but has meaning to GSI. The common name (CN) portion of the DN is expected by the client to contain the fully-qualified domain name (FQDN) of the host on which the server resides, prefixed by a service-specific tag.

Services running as root (e.g. the Globus Gatekeeper and GridFTPD) typically share a Host Credential which uses a prefix of "host/". For example a Globus Gatekeeper (described later in Section 3.1) on the host "foo.mcs.anl.gov" would be expected to have a CN of "host/foo.mcs.anl.gov".

The Globus Monitoring and Discovery Service uses its own credential with a prefix of "ldap/".

2.3.3 Credential Acquisition

As mentioned above, certificates are acquired from trusted parties known as certificate authorities (CAs). Certificate acquisition in GSI is basically the same as X.509 certificate acquisition in any PKI.

A user would use the grid-cert-request tool from the Globus Toolkit to acquire credentials. The tool starts by creating a set of cryptographic keys - one private and one public. The private key, as its name implies, is kept secret by the user, who has sole knowledge of this key. The public key is used to form a certificate request

which is sent to a CA of the user's choice. That CA will have a process it follows to establish the user's identity, which it will embed, along with the public key, in a certificate it returns to the user.

A service administrator also uses the `grid-cert-request` tool to acquire credentials for a service. The primary difference is that the administrator is getting credentials not for themselves but for the service, so the CA will establish the identity of the service (as explained in the previous section on Credential Identity) and verify the administrator is authorized by the CA's policy to administer the service in question.

The Globus Project operates a research CA which signs certificates for Globus Toolkit users, however the use of this CA is being depreciated as more production CAs are being setup by various organizations like DoE, the European Data Grid, NCSA, NPACI, etc.

Since long-term credentials are typically valid for a period of years, this step of acquiring long-term credentials is not a frequent activity.

2.3.4 Credential Storage

GSI credentials are typically stored on local disk. User' credentials are typically in their home directory ("`~/globus/usercert.pem`" and "`~/globus/userkey.pem`" to be precise). The private key is normally protected by being encrypted with a pass phrase known only to the user in addition to using filesystem permissions.

Services in GSI typically have their long-term credentials stored on local disk, protected by just filesystem. The private key is typically not encrypted since it's administratively difficult to have someone present to decrypt it everytime a service starts up.

GSI also supports using hardware tokens using the PKCS11 interface. [I doubt this is documented anywhere, but can be enabled with a compile-time definition. – VW]

2.3.5 Certificate Revocation

Since user and service certificates are valid for a long period of time, the issue of compromise of the credentials (e.g. lost or stolen private key) must be addressed. A standard revoking X.509 certificates exists called a certificate revocation list (CRL).

A CRL is a list, issued by a CA, of all the certificates that it has issued that are believed to have been compromised and no longer should be trusted for authentication. The CRL is signed by the CA to prevent modification.

GSI supports CRLs, but makes them optional as they tended to be difficult to manage in practice and not all sites will choose to implement them. If a CRL is present for a CA, then GSI will use it and fail authentication for any certificates present in it. If a CRL is not present, GSI will operate as if it were an empty list.

For authentication using GSI proxy credentials, the actual user credential that was used to create the proxy credential is checked against the CRL. Proxy certificates themselves, because of their short lifetime, are not revoked.

Details about GSI support of CRLs can be found at:

<http://www.globus.org/security/v2.0/crls.html>

2.4 Proxy credentials

A Globus user normally does not use these long-term credentials directly but instead uses them to create a proxy credential and then uses the proxy credential to authenticate. As shown in step 2 of Figure 1 the user does this using the `grid-proxy-init` tool along with their long-term credentials and the pass phrase to decrypt their private key. This step is analogous to “logging onto the grid.”

Proxy credentials are a GSI extension to standard X.509 credentials. In summary they are a short-lived (typically hours) set of credentials that allow the user to assert the same identity as their long-term credentials. Because they are short-lived, the private key of the proxy credentials is not required to be encrypted as the private key of the long-term credentials is. This means that applications run by the user can easily use the proxy credentials without having to get a pass phrase from the user to decrypt them. (For those familiar with Kerberos, the proxy credentials can be thought of as analogous to the Kerberos ticket cache.)

Since the proxy credential normally has a lifetime of just hours, this process of running `grid-proxy-init` tends to be done by users on a daily basis.

2.4.1 Motivation for Proxy Credentials

The motivation for proxy credentials is to support single-sign on and delegation. Single-sign on is the act of authenticating once and then being able to initiate multiple actions based off of that single authentication. Delegation is the ability to temporarily grant some of their rights to a remote process.

Single-sign on is important since Grid computing often involves orchestrating complex tasks involving multiple resources requiring the user to repeatedly authenticate themselves. and having to authenticate repeatedly would be a prohibited burden. Since the security considerations of user’s long-term private key requires that it encrypted, the user would be forced to repeatedly type their pass phrase repeatedly, creating an prohibitive burden. While it would be possible to cache the pass phrase in memory, this creates it’s own set of security risks.

Delegation, which is described in detail in Section 2.7, is important since a user’s process may need to interact with other resources. For example a computational job may need to retrieve input data from a storage system and possibly also to store the results of the computation. Having the user be present to authenticate these interactions may not be possible since jobs may be have very long lifetimes. Resources on the Grid may have no trust of each other either since they may be in different administrative domains. This means the user needs to be able to grant to their process to act on their behalf.

2.4.2 Proxy Credential Creation

As shown in Figure 1, to generate a set of Proxy Credentials, a user runs a program called `grid-proxy-init` (part of the Globus Toolkit).

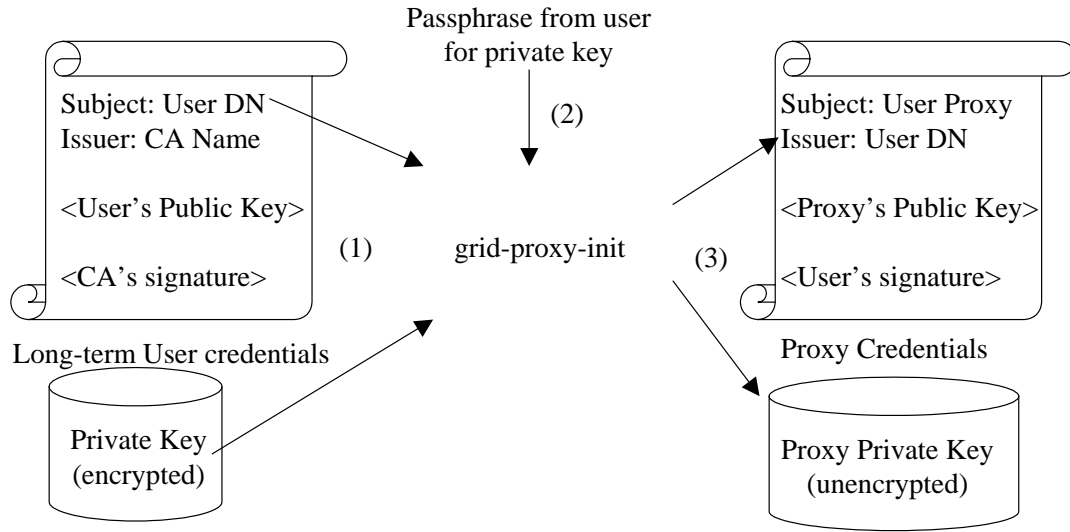


Figure 4: GSI Proxy Credential Creation

Grid-proxy-init performs the following steps as shown in the Figure:

1. It reads the user's long-term credentials from their home directory.
2. It then prompts the user for their pass phrase to decrypt their long-term private key.
3. It then generates a new key public and private key, signs the new public key with the user's long-term private key to generate the proxy certificate and then writes the proxy credentials out to a proxy file.

2.4.3 Proxy Credential Storage

Proxy credentials are typically stored in a local scratch filesystem. On most Unix hosts this is /tmp. The filename varies, but is usually based off of the user's numeric user id.

The short lifetime of the Proxy Credentials allow the security considerations of the private key to be more relaxed since it has less time to be compromised. Because of this it is protected with just normal filesystem permissions instead of being encrypted. This allows the Proxy Credential to be easily used repeatedly by the user and the user's applications to authenticate the user until the Proxy Credentials either expire or are manually destroyed by the user.

2.4.4 Full versus Limited Proxy Credentials

GSI currently offers two forms of Proxy Credentials – Full and Limited. A Full Proxy Credential grants the bearer all of the issuing users rights. A Limited Proxy grants all of the user's normal rights except for process creation rights. For example a Limited Proxy Credential could not be used to initiate a computation job via the Globus Gatkeeper (as described in Section 3.1).

A more fine-grain delegation is currently under development and will be included in a future release of GSI.

2.4.5 Proxy Credential Format

Proxy Credentials are a new certificate and private key created by the user. The certificate is actually signed by the user’s long-term private key. Unlike the user’s long-term certificate, the certificate in the Proxy Credentials had a typical lifetime on the order of hours.

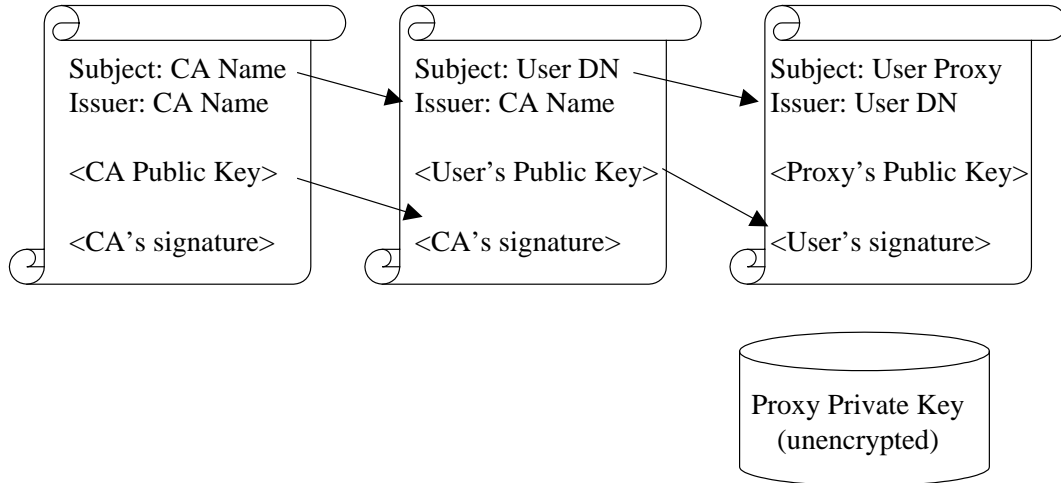


Figure 5: A GSI Proxy Credential

A GSI proxy credential is shown in detail in Figure 5. This shows the actual Proxy Certificate and private key on the right, the long-term user certificate that signed the proxy in the middle, and the CA certificate on the left.

The important thing to notice from this figure is what is called the certificate chain. Each certificate is signed by the certificate preceding it, creating a chain of trust from the CA all the way to the proxy. A relying party can verify the user credential was issued by the CA, and allows them to check the proxy credential was issued by the user credential.

2.5 Establishing a GSI-secured connection

Once a user has created a proxy credential they can use that credential to run GSI-enabled applications (e.g. a gridftp client, globusrun or other tools that come with the Globus Toolkit) as shown in step 3 of Figure 1 and step 2 of Figure 2.

The client and server applications connect using a normal TCP socket. Then the applications both make calls to the GSI libraries through the Generic Security Services API (GSS-API) [XXX]. By invoking GSS-API calls, the application uses GSI to set up a SSL-secured connection to a server.

The GSI libraries use the OpenSSL [XXX] library to perform all it’s cryptographic operations and to actually implement the SSL protocol. This library was chosen due to it’s the fact it’s open source and popular, which has led it to be highly scrutinized by the community for security flaws.

As part of setting up the SSL-secured connection the client and server both authenticate themselves to each other, the client using the user's proxy credentials and the server using its credentials.

During authentication, both the client and server perform the following checks::

1. The remote party's credentials must be signed by a certificate authority (CA) that are trusted by the local side. A list of trusted CAs is maintained on the local disk (see Section 2.8.1 for more details).
2. If a Certificate Revocation List (CRL) is present for the CA that issued the remote party's credentials, is present on the user's local disk, it is checked and the certificate for the server must not appear in the revocation list. GSI's support of CRLs is discussed in more detail in Section 2.3.5.

Once GSI authentication is complete, the result is the identity of the remote party is available to the local party to make authorization decisions and GSI can be used to protect any further user data sent between the client and server. Protection options include any message protection normally offered by the SSL protocol, including integrity protection and encryption.

2.6 Authorization

After a remote party has authenticated themselves, the next step is to perform authorization. Authorization is the act of deciding if the authenticated party is allowed to access a resource (from the server perspective) or is the appropriate service (from the client perspective).

Both the GSI client and server side perform an authorization check based on the identity of the remote party returned by the GSI authentication. In this section we described both of those checks.

2.6.1 Client-side Authorization

When a GSI-enabled client connects to a server, it expects the identity of server to be based on a service-specific prefix concatenated with the fully-qualified domain name (FQDN) of the host on which the service resides. If the service identity does not meet this criteria, the GSI client library returns an error to the application which typically aborts the connection.

For example a Globus Gatekeeper (described later in Section 3.1) on the host "foo.mcs.anl.gov" would be expected to have a CN of "host/foo.mcs.anl.gov".

A client application can override this default behavior by supplying an expected identity to the GSI client libraries.

2.6.2 Server-side Authorization

The Globus Toolkit comes with a simple server-side authorization mechanism, which uses the "grid-mapfile." The grid-mapfile is the main point of control for the local system administrator to control who can use their system using Globus.

The grid-mapfile is a plain text file containing a list of authorized GSI identities and mappings from those GSI identities to local user identities (e.g., Unix account

names). Thus it is both an access control list and a mapping mechanism. Entries are added to it by the local system administrator, when requested by a user who presents their GSI identity to the administrator. The administrator then determines what the local account name is for the user and adds this mapping to the grid-mapfile.

An example mapfile is shown here. This grid-mapfile has two entries, indicating the two users whose GSI identities appear in the left column are authorized to use the local resource and should map to the Unix usernames given in the right column.

```
# Sample grid-mapfile
# GSI Identity                               Unix Username
"/O=Grid/O=Globus/OU=mcs.anl.gov/CN=Joe User"    joe
"/O=Grid/O=Globus/OU=mcs.anl.gov/CN=Jane User"    jane
```

As shown in step 4 of Figure 2, the grid-mapfile is used by GSI after authenticating a user. It first checks to see if the user's Grid identity is listed in the grid-mapfile. If the user is not listed they are denied access to the resource.

If they are listed in the grid-mapfile, GSI gets the name of their local Unix user account from the grid-mapfile, changes to the local identity (i.e., performs a `setuid()` system call) and then runs the task requested by the user.

Note that on a Unix system even if a user is in the grid-mapfile, they still must be in `/etc/passwd`. The grid-mapfile does not replace `/etc/passwd` but layers on top of it.

2.7 GSI Delegation

A common requirement for Grid jobs is the job to be able to access other resources on the user's behalf. For example, a computational job may need to contact a remote storage system to retrieve an input data set or store results. This storage system may be in a different administrative domain and have knowledge of the user, but no implicit trust of the resource the job is running on.

The solution to this is to allow the user to delegate rights to jobs they start remotely in order to allow those jobs to act on the user's behalf. GSI enables this by allowing a user to create a proxy credential remotely over a GSI-protected connection.

Step 4 of Figure 1 shows the optional step of delegation. If so requested by the user and client application the GSI libraries can delegate a proxy credential (described in Section 2.4) to the server. The proxy credential will be handed off to any process started on the user's behalf. This delegated proxy credential can then be used by that process to perform GSI authentication with other GSI-enabled Grid resources.

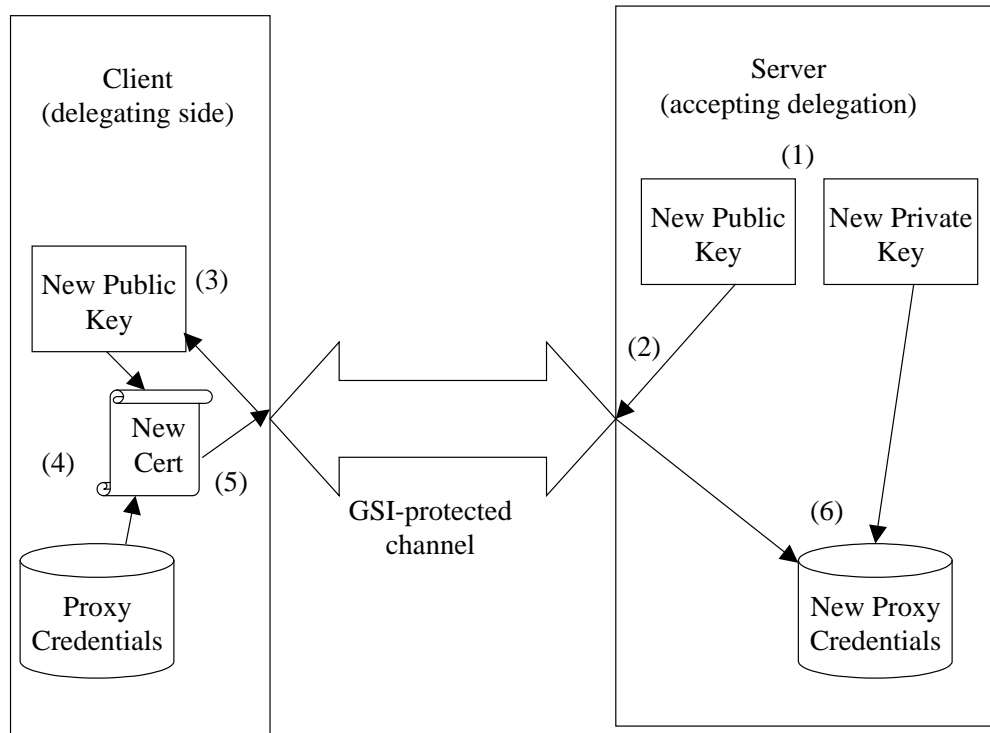


Figure 6: GSI Delegation

Figure 6 shows GSI's delegation process. This process is same process performed locally by `grid-proxy-init`, as described in Section 2.4.2, but is done over a GSI-protected channel. On the left is the client, which already has a GSI proxy. On the right is a server receiving the delegation from the client. The steps of the delegation process are:

1. The server generates a new pair of public and private keys.
2. The server sends public key over the GSI-protected channel to the client.
3. The client receives the new public key.
4. The client now signs the public key using the private key from it's proxy credentials creating a new proxy certificate.
5. The client sends the new proxy certificate along with all of it's the certificates (but not it's privat key) back to the server. It sends along all of it's own certificates so that the proxy on the server will have all the certificates required to create the full certificate chain back to the certificate authority.
6. The server receives the certificates from the client and joins them with the private key it generated in step 1 to create a new set of proxy credentials.

One thing to noticed about the delegation process is that at no point do any private keys, or any non-public information, pass over the network. Only certificates, which are public information, and the generated public key from the server. This means that

encryption is not necessary on the channel. However since the public key could be tampered with, integrity checking is required to make sure it stays unmodified.

GSI also offers the option of delegating a Limited Proxy. A Limited Proxy, as described in Section 2.4.4, grants all of the user's normal rights except for process creation rights.

2.8 GSI Configuration

In this Section we described the configuration available to administrators of the GSI services and users of GSI in order to control the authentication and authorization. GSI configuration is typically stored in a local directory, usually `/etc/grid-security` on Unix systems, and is used by all GSI-enabled applications on the system. However, as described in the following subsections, it can be located elsewhere and tailored for individual services and users.

2.8.1 Trusted Certificate Authorities

The trusted certificates directory contains all the certificates of certificate authorities (CAs) that the local system trusts to issue long-term credentials as described in Section 2.3.

Each CA also has a signing policy included with it. This is a file that specifies the namespace the CA is allowed to issue certificates in. Typically this is used to allow CAs to only issue certificates with identities that start with a certain prefix. This allows a resource admin to make sure that two different CAs don't both issue a certificate with the same identity.

Also, as described in Section 2.3.5, each CA may optionally have a certificate revocation list (CRL). The CRL for a CA would appear in the trusted certificates directory.

The trusted certificates directory is normally in `/etc/grid-security/certificates`, but can be specified with the `X509_CERT_DIR` environment variable.

2.8.2 Grid-mapfile

The use of the grid-mapfile is described in more detail in Section 2.6.2, but this file basically controls what GSI identities have the right to use the local resource and their local identity (e.g. Unix user name) is when they do so.

This file is normally located at `/etc/grid-security/grid-mapfile` but this may be overridden with the `GRIDMAP` environment variable.

3 Globus Toolkit Services

In this section we describe the three main higher-level services that come with the Globus Toolkit. These services are:

- The Gatekeeper and Job manager, for allowing job submission and control
- GridFTP for moving files on and off a resource

- The Monitoring and Discovery Service (MDS) for information discovery about a resource

3.1 Gatekeeper and Job Manager

While the Gatekeeper and Job Manager are actually two separate things, it is generally simpler to think of them together as the single mechanism used by users to start computational jobs on a resource.

The Gatekeeper is analogous to a secure inetd service. Like inetd, it demuxes user connections based on the service they desire, but it also has the role of authenticating, authorizing and setting user privileges (e.g., running `setuid()`). It is normally run from inetd as root. After authenticating the user using GSI (as described in Section 2), it verifies the user is authorized to access the system, and what account they should run under, by checking the grid-mapfile (as described in Section 2.6.2). If either the authentication or authorization fails, the connection is dropped.

After authenticating and authorizing the user, the gatekeeper receives the name of the particular service the user wishes to invoke (with inetd, this would come from the port number the user connected to; with the Gatekeeper this is passed in-band). The gatekeeper proceeds to invoke the service. First, it does a call `setuid()` system call to change to the user's local identity as defined by the system administrator in the grid-mapfile as described in Section 2.6.2. Then it executes the requested service.

The only service normally supported by the Gatekeeper is the Job Manager. Additional services may be added by adding entries for them in the `$GLOBUS_LOCAION/etc/grid-services` directory.

After the Job Manager is started, it receives the details of the job submission request from the user, starts the job, and provides an interface for the user to monitor and control (e.g., kill) the job. If a local queuing system (e.g., PBS, LSF) is already present on the resource, the job manager can interface with it, using the standard clients for the queuing system to submit and control the job. Otherwise the job manager will simply spawn a child process to run the job.

It is worth noting that since the Job Manager is started under the user's identity (because the gatekeeper has done a `setuid()` before starting it), it is not privileged and so it is limited by standard file permissions and operating system privileges to operations the user could normally do.

More information about the Gatekeeper and Job Manager may be found at [9].

3.2 GridFTP

GridFTPD allows Globus users to move files onto and off of a system. It is a standard ftp server implementing a standard FTP protocol with extensions for security using GSI as defined by [10] and high performance as defined by the GridFTP working group of the Global Grid Forum [11].

GridFTPD is normally run from inetd as root. It accepts connections, authenticates the connection using GSI, checks authorization using the grid-mapfile, logs the event

to syslog, switches mode to the user as defined in the grid-mapfile, and then presents a standard ftp interface to the user.

More information about GridFTP may be found at [12].

3.3 Monitoring and Discovery Service

The Monitoring and Discovery Service (MDS) is a distributed information system. This service allows users to obtain basic information about a system, such as operating system type, number of CPUs, amount of memory, system load, etc.

Access to MDS is by default anonymous and unauthenticated, but may be configured to perform authentication and uses the a authorization system based on it's own copy of the grid-mapfile described in Section 2.6.2.

More information about MDS can be found in [13] and [14]. Note that MDS was also known as the Metacomputing Directory Service and the Meta Directory Service.

4 Firewall Traversal

[This section is pretty sparse at the moment. I have a firewall testbed set up and will be flushing it out once I get a chance to do some testing. – VW]

[Should cite Firewalls experiences document from Portsmouth, UK found at <http://esc.dl.ac.uk/Papers/firewalls/globus-firewall-experiences.pdf> - VW]

[Some figures might also help this section, especially the NAT part – VW]

Firewalls are network devices designed to protect machines from inappropriate access. This is usually done by limiting access from the outside to selected well-known ports. This causes problems with tools like the Globus Toolkit that dynamically create processes that need to communicate over system-assigned port numbers.

The Globus Toolkit provides a way to restrict which ports its client and server programs use. This is done by the use of the GLOBUS_TCP_PORT_RANGE environment variable. Setting this variable to a range of ports will cause Globus tools to use only ports in this range.

The table in Appendix 1 lists all the ports used by the Globus Toolkit.

4.1 Network Address Translation

The use of Network Address Translation (NAT) firewalls creates a number of problems for Grid software.

First, for servers behind it NAT firewall it masks the server's true identity. This means that when a client is connecting to such a server, it believes that it is talking to a host with the IP address and hostname of the firewall instead of the server. And since GSI software uses this hostname as the basis for the expected identity of the server, this will cause failures with unexpected identities.

Second, resources will have different hostnames than so perceived by the outside world. Often a process will want to establish a incoming port to accept connections

from clients or peer applications and then communicate this port to those other applications. However, the application on the resource behind a NAT firewall will give out it's local resource hostname which is not routable through the firewall.

Third, all traffic that can pass inbound through a NAT firewall must be configured by the firewall administrator. This makes dynamically creation of new ports for inbound connections impossible.

Because of this we do not believe it is possible to reasonably run Globus service behind a NAT firewall at this time. We are working on methods to correct this and expect a future version of the toolkit to allow this.

[Clients are less certain. Testing this now – more details to appear soon. – VW]

5 Using Globus with Kerberos

As described in this Section, Globus and Kerberos may be used together in two ways: by mapping between GSI and Kerberos and by replacing GSI with Kerberos. We also explain why mapping between GSI and Kerberos is the preferred method.

5.1 Mapping between GSI and Kerberos

Mapping between GSI and Kerberos involves using credential translators, special applications that can use credentials from one mechanism to acquire credentials for another, to convert between Kerberos and GSI at appropriate points. This gives the appearance to the user of using only one security mechanism (either GSI or Kerberos as desired), while both are actually being used under the covers. All the Globus Toolkit continues to use GSI as it's authentication mechanism.

In this section we cover the two cases of mapping: using SSLK5 to translate from GSI credentials to Kerberos credentials when a user enters a Kerberos domain from the Grid and using KX509 to translate from Kerberos credentials from GSI credentials for user in a Kerberos domain connects to the Grid.

5.1.1 Mapping from GSI to Kerberos

Mapping from GSI to Kerberos is normally done when a Grid user, with GSI credentials, authenticates to a host in a Kerberos domain. As shown in Figure 7, (1) a user would authenticate with and delegate GSI credentials to the resource. The GSI credentials would then be used through (2) SSLK5, a GSI-to-Kerberos credential translator described in detail later in this section, to acquire Kerberos credentials. The user would then have both Kerberos and GSI credentials available on the resource and (3) use the Kerberos credentials to connect to Kerberos resource in the local realm or (4) use the GSI credentials to connect to other Grid resource.

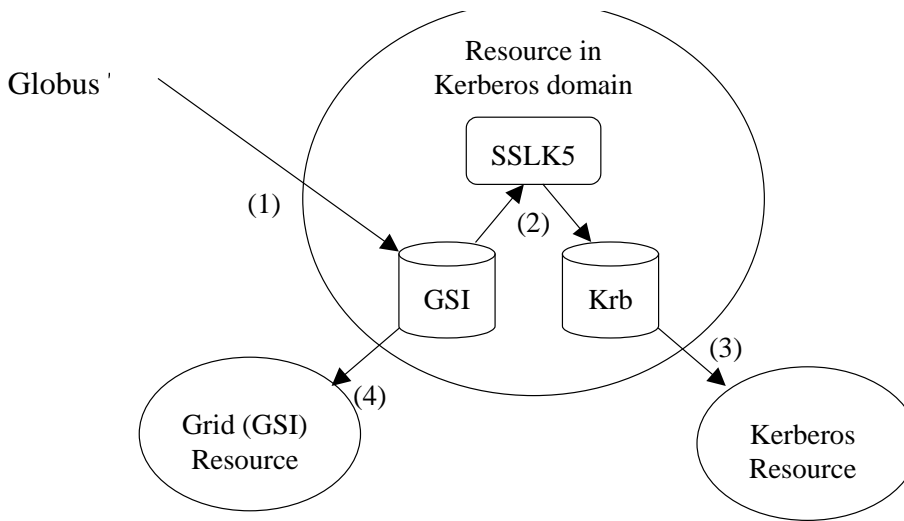


Figure 7: Mapping from GSI to/from Kerberos

5.1.2 SSLK5 Details

SSLK5 and its accompanying daemon program SSLK5d use a user's existing GSI proxy credentials to acquire Kerberos credentials. This is normally used at a site that has an existing Kerberos installation and is accepting connections from users authenticating with GSI. This site wishes to security grant these users Kerberos credentials based on their GSI credentials so they can seamlessly using existing Kerberos services. By using these tools users connecting with GSI can automatically acquire Kerberos credentials.

SSLK5 accomplishes this by having the SSLK5 daemon run on the site's Kerberos Domain controller (KDC) where it has access to the Kerberos database and can generate Kerberos credentials. A local configuration is maintained in a mapping file that maps GSI identities to Kerberos identities (principal names).

SSLK5 works as shown in Figure 8:

1. A user runs SSLK5, which uses the user's GSI credentials to authenticate to a SSLK5D server and establish a GSI-protected channel.
2. The SSLK5D checks for the user's GSI identity in its configuration and maps it to a Kerberos identity.
3. Using the user's Kerberos identity and information from the Kerberos database SSLK5D builds a set of Kerberos credentials for the user.
4. SSLK5D sends the Kerberos credentials back to the SSLK5 client which stores them in the user's default Kerberos credential cache.
5. Normal Kerberized applications can then use the acquired Kerberos credentials as they normally would.

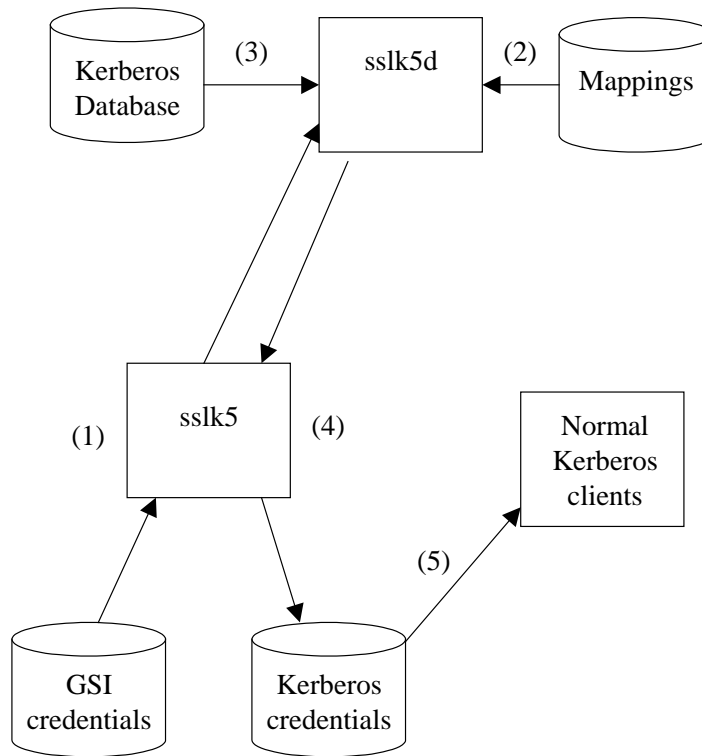


Figure 8: Acquiring Kerberos credentials from GSI credentials using SSLK5

5.1.3 Mapping from Kerberos to GSI

Mapping from Kerberos to GSI is normally done when a Kerberos user wishes to use standard Grid resources. As shown in Figure 9, (1) a user with Kerberos credentials would use those credentials and KCA, a GSI to Kerberos credential application described in detail later in this section, to (2) obtain a set of GSI credentials. The user would then use those GSI credentials to run Grid applications as normal.

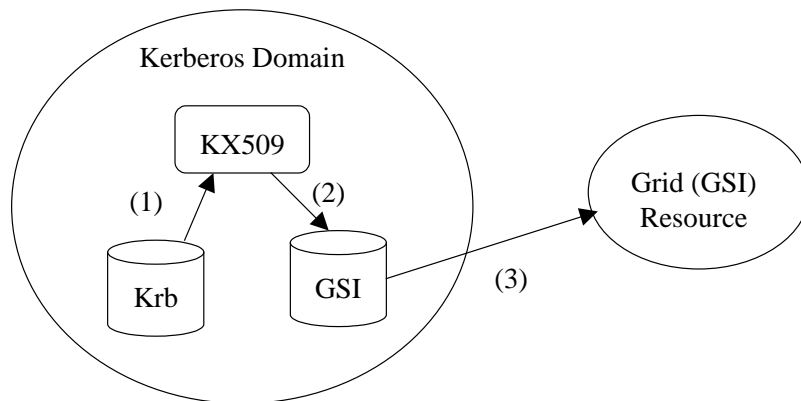


Figure 9: Kerberos to GSI Mapping

5.1.4 KX509 Details

KCA is basically an online CA that users use the KX509 client to connect to and receive a GSI proxy. Figure 10 shows how KX509 and KCA work to translate Kerberos credentials to GSI credentials.

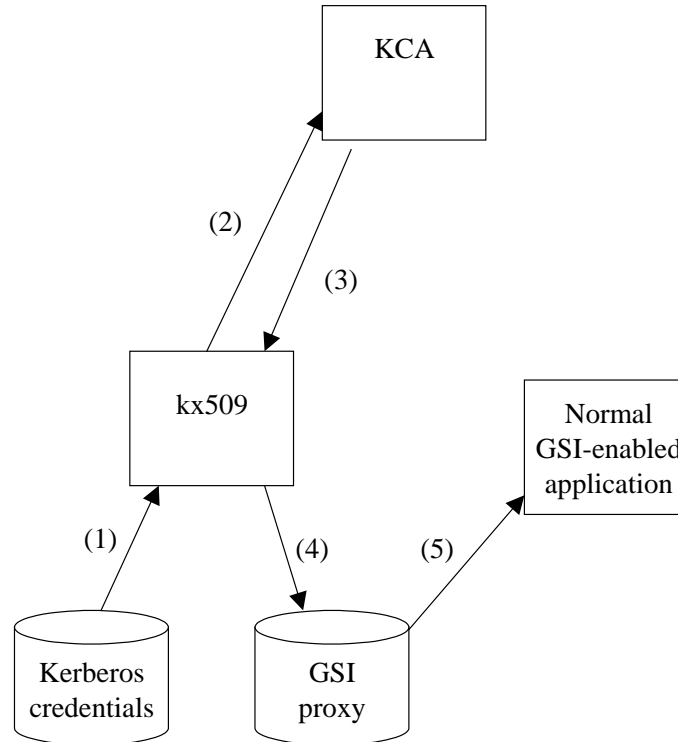


Figure 10: Use of kx509/kca to translate from Kerberos to GSI

The steps shown in Figure 10 are:

1. A user with existing Kerberos credentials (i.e. a TGT stored in a local ticket cache) runs kx509
2. Using the Kerberos credentials kx509 connects to a KCA server and authenticates.
3. KCA creates a GSI proxy – a certificate it signs with an identity based off of the user's Kerberos principal name and returns this to the kx509 client.
4. The kx509 client stores this GSI proxy in a standard GSI proxy file.
5. GSI applications can now use the proxy as normal.

5.2 Replacing GSI with Kerberos

A less ideal method to credential mapping is to remove the GSI libraries from the Globus Toolkit software stack and replace it with Kerberos libraries. This causes the Toolkit itself to use Kerberos credentials without any form of translators so that users with Kerberos credentials can use it directly.

However the resulting tools and installations are incompatible with normal GSI-based Globus installations and tools. For this reason, we highly recommend mapping instead.

[Be nice if I had a web page to reference here with more information – VW]

5.3 Using Globus with the Andrew File System (AFS)

A credential translator exists for converting GSI credentials to AFS tokens called GSIKLOG [XXX]. GSIKLOG works the similarly to SSLK5 described in Section 5.1.1. A service running on a host with AFS would run GSIKLOG automatically for users, acquiring an AFS token for them. This would allow users of GridFTP to access files stored in AFS or jobs submitted through the Gatekeeper to access files into AFS.

XXX More? Where do I find out how to set this up?

[Yep, need a web page on this – VW]

<ftp://achilles.ctd.anl.gov/pub/DEE/gssklog-0.1.tar>

6 Frequently Asked Questions

6.1 Does putting Globus on a system mean anyone can use it?

No. Like installing sshd or ftpd, installing Globus on a system only adds a standard set of ways authorized users can access the system. The resource administrator still controls the list of users who can access the system through */etc/passwd* and the *grid-mapfile* as described in Section 2.6.2, as well as by the CA policy file

6.2 How do I know Globus is secure?

Globus strives to maintain security through using well-tested publicly-available code whenever possible, namely the OpenSSL libraries, and making our own code relatively small and open source, so that it can be reviewed by anyone who desires to do so.

We note that Globus Toolkit code has been reviewed from a security perspective by a variety of site security officers.

6.3 If I use Globus, aren't I dependant on the quality of other people's CAs?

As described in Section 2.8.1 you control the list of CAs that you will accept certificates from. This allows you to investigate each CA, determine if you are comfortable with its methods and security and accept or deny certificates for each CA on a CA-by-CA basis.

7 Summary

We have provided an overview of the Globus Toolkit and the Grid Security Infrastructure (GSI) aimed at system administrators and site policy decision makers. We have described how the Globus Toolkit and GSI layer on top of existing software and explained how the GSI works and the mechanisms available to an administrator to control who is authorized to access their resources.

We have also described the higher-level services that come with the Globus Toolkit and how each interacts with the resource. Finally we answered a number of commonly-asked questions about the Globus Toolkit.

For those seeking greater detail we recommend the Security section of the Globus Web Site[15], the web page for the GSI Working Group in the Global Grid Forum [16] and the following literature [17-19].

Questions may also be posed to the Globus Project and its community on the open email discussion forum by sending email to discuss@globus.org.

References

[This probably needs to be updated – VW]

1. Foster, I., C. Kesselman, and S. Tuecke, The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 2001. **15**(3): p. 200-222.
2. Johnston, W.E., D. Gannon, and B. Nitzberg. Grids as Production Computing Environments: The Engineering Aspects of NASA's Information Power Grid. in *Proc. 8th IEEE Symposium on High Performance Distributed Computing*. 1999: IEEE Press.
3. National Computation Science Alliance. 2001: <http://alliance.ncsa.edu/>.
4. National Partnership for Advanced Computational Infrastructure. 2001: <http://www.npaci.edu/>.
5. DOE Science Grid. 2002: <http://www.doesciencegrid.org/>.
6. Arsenault, A. and S. Turner, Internet X.509 Public Key Infrastructure, PKIX Roadmap. 2000.
7. Dierks, T. and C. Allen, The TLS Protocol, Version 1.0. 1999.
8. OpenSSL Project. 2002: <http://www.openssl.org>.
9. Globus Resource Access Manager. 1997.
10. Horowitz, M., FTP Security Extensions, in *INTERNET RFC 22228*. 1997.
11. Global Grid Fourm GridFTP Working Group. 2002: http://www.gridforum.org/6_DATA/data.htm.
12. GridFTP Protocol and Software. 1997.
13. Czajkowski, K., et al. Grid Information Services for Distributed Resource Sharing. in *10th IEEE International Symposium on High Performance Distributed Computing*. 2001: IEEE Press.
14. Metacomputing Directory Service (MDS). 1997.

15. Globus Security Web. 2002: <http://www.globus.org/security/>.
16. Global Grid Forum Grid Security Infrastructure Working Group. 2002: <http://www.gridforum.org/security/>.
17. Foster, I., et al. A Security Architecture for Computational Grids. in *Proceedings of the 5th ACM Conference on Computer and Communications Security*. 1998.
18. Butler, R., et al., Design and Deployment of a National-Scale Authentication Infrastructure. *IEEE Computer*, 2000. **33**(12): p. 60-66.
19. Novotny, J., S. Tuecke, and V. Welch. An Online Credential Repository for the Grid: MyProxy. in *10th IEEE International Symposium on High Performance Distributed Computing*. 2001: IEEE Press.

Appendices

1 Globus Toolkit Port Usage

[Should be appendix A – VW]

Application	Globus Version	Network Ports	Network Addresses	Comments
Gatekeeper	1.1.3 and later	2219/tcp	From client machines to hosts running Globus gatekeepers. Direction of traffic depends on location of client and server.	Port defined by IANA
MDS Grid Resource Information Service (GRIS)	1.1.3 and later	2135/tcp 2135/udp	To hosts running a GRIS service (typically all machines that run the Globus services)	Ports defined by IANA
MDS Grid Information Index Service (GIIS)	1.1.3 and later	Site-selected	To hosts running a GIIS service (typically a small number of machines that index information from multiple GRIS services for searching purposes)	Each site defines its own GIIS hosts and port numbers.
GridFTP	All	2811/tcp for control channel. ¹	To hosts running GridFTP server	Port defined by IANA
GSI-Enabled SSH	All	22/tcp	To hosts running GSI-Enabled SSH server.	Same as standard ssh.
MyProxy	All	7512/tcp	To hosts running MyProxy server	Default. Can be modified by site.

¹ Data channel port numbers are random, but can be limited by the GLOBUS_TCP_PORT_RANGE environment variable

2 Overview of Standard Technologies on which GSI is based

[Should be Appendix V – VW]

[This section will eventually contain basic introductions to all the technologies, right now it's just some text for starters. – VW]

2.1 PKI

A Public Key Infrastructure (PKI) is an infrastructure for authentication based on trusted third parties known as Certificate Authorities (CAs). CAs exist solely to assign identities to entities (e.g. users, hosts, services, etc.) and distribute X.509 Certificates to those entities. Some examples of commercial CAs are Thawte (<http://www.thawte.com/>) and Verisign (<http://www.verisign.com/>).

Every entity in a PKI has a certificate and a public and private key pair. The certificate contains the entities identity and their public key. It is signed by a CA and serves to bind the identity to the public key. While the certificate is public the private key is known only to the entity.

Possession of a certificate and accompanying private key allows an entity to assert their identity by presenting their certificate and proving possession of their private key. This provides a mechanism for authentication to anyone who trusts the CA that issued their certificate.

Each CA has a mechanism they use for issuing determining an entity's identity and issuing them a certificate. This information, allow with other details on how the CA is run, is normally published in what is known as a Certificate Policy. This is the document one would normally look at when one is deciding whether or not one can trust certificates issued by a given CA for authentication.

2.2 X.509 Certificates

These credentials created by the user in conjunction with a trusted party known as a certificate authority (CA):

1. The user first creates a pair of cryptographic keys using a tool like grid-cert-request, which comes with the Globus Toolkit. This pair of keys consists of a private and public key.
2. The private key is encrypted with a pass phrase of the user's choosing and stored in the user's home directory.
3. The public key is sent to a CA of the user's choice.
4. The CA will perform some procedures to identify the user and then issue the user a certificate. This certificate is basically a binding of the identity of the user and the user's public key that is cryptographically signed by the CA.
5. The CA then gives this certificate back to the user, who stores it along-side their private key.

2.3 SSL

Secure Socket Layer (SSL), or Transport Layer Security (TLS) as it is known in current IETF standards activities, is a standard method for providing authentication, privacy (encryption) and integrity (secure checksumming) over a TCP connection. SSL can claim to be the most widely-used security protocol today due to its deployment as the mechanism for secure web transactions (https).

Entities using SSL use Public Key Infrastructure X.509 Certificates to authenticate themselves (as described in the previous subsection **Error! Reference source not found.**). SSL provides the option for one or both parties of a connection to authenticate themselves. In the Web world, typically only servers authenticate themselves. However, as we will describe later in this paper, the Grid Security Infrastructure performs mutual authentication: both the client and the server authenticate themselves.