Globus Toolkit 6.0 Key Concepts

DRAFT

Table of Contents

DRAFT

GT 6.0 GridFTP Key Concepts

DRAFT

GT 6.0 GridFTP Key Concepts

Overview

One of the foundational issues in HPC computing is the ability to move large (multi Gigabyte, and even Terabyte), file-based data sets between sites. Simple file transfer mechanisms such as FTP and SCP are not sufficient either from a reliability or performance perspective. GridFTP extends the standard FTP protocol to provide a high-performance, secure, reliable protocol for bulk data transfer.

DRAFT

Table of Contents

DRAFT

Chapter 1. GridFTP Protocol

GridFTP is a protocol defined by Global Grid Forum Recommendation GFD.020, RFC 959, RFC 2228, RFC 2389, and a draft before the IETF FTP working group. Key features include:

- Performance - GridFTP protocol supports using parallel TCP streams and multi-node transfers to achieve high performance.
- Checkpointing - GridFTP protocol requires that the server send restart markers (checkpoint) to the client.
- Third-party transfers - The FTP protocol on which GridFTP is based separates control and data channels, enabling third-party transfers, that is, the transfer of data between two end hosts, mediated by a third host.
- Security - Provides strong security on both control and data channels. Control channel is encrypted by default. Data channel is authenticated by default with optional integrity protection and encryption.

DRAFT

Chapter 2. Globus Implementation of GridFTP

The GridFTP protocol provides for the secure, robust, fast and efficient transfer of (especially bulk) data. The Globus Toolkit provides the most commonly used implementation of that protocol, though others do exist (primarily tied to proprietary internal systems).

The Globus Toolkit provides:

- a server implementation called `globus-gridftp-server`,
- a scriptable command line client called `globus-url-copy`, and
- a set of development libraries for custom clients.

While the Globus Toolkit does not provide a client with Graphical User Interface (GUI), Globus Online[1] provides a web GUI for GridFTP data movement.

Globus GridFTP framework implements all the key features of GridFTP protocol mentioned above. It supports both Grid Security Infrastructure (GSI) and SSH for securing the data transfer. Unlike sftp, SSH based GridFTP supports multiple security options on the data channel - authentication only, authentication and integrity protection, fully encrypted. Globus implemention of GridFTP is modular and extensibl. XIO based Globus GridFTP framework makes it easy to plugin alternate transport protocols. The Data Storage Interface (DSI) allows for easier integration with various storage systems. We currently have DSIs for POSIX filesystems (default) and HPSS. Globus GridFTP has been deployed at thousands of sites with more than 10 million data transfers per day.

DRAFT

[1] http://www.globusonline.org

Chapter 3. GridFTP Clients

Globus Online[1] is the recommended interface to move data to and from GridFTP servers. Globus Online provides a web GUI, command line interface and a REST API for GridFTP data movement. It provides automatic fault recovery and automatic tuning of optimization parameters to achieve high performance.

The Globus Toolkit provides a GridFTP client called `globus-url-copy`, a command line interface, suitable for scripting. For example, the following command:

```
globus-url-copy gsiftp://remote.host.edu/path/to/file file:///path/on/local/host
```

would transfer a file from a remote host to the locally accessible path specified in the second URL.

Finally, if you wish to add access to files stored behind GridFTP servers, or you need custom client functionality, you can use our very powerful client library to develop custom client functionality.

For more information about GridFTP, see:

- the documentation.
- The Globus Striped GridFTP Framework and Server[2]

DRAFT

[1] http://www.globusonline.org

GT 6.0 GRAM5 Key Concepts

DRAFT

GT 6.0 GRAM5 Key Concepts

Overview

The Globus Toolkit provides GRAM5: a service to submit, monitor, and cancel jobs on Grid computing resources. In GRAM, a job consists of a computation and, optionally, file transfer and management operations related to the computation. Some users, particularly interactive ones, benefit from accessing output data files as the job is running. Monitoring consists of querying and/or subscribing for status information, such as job state changes.

Grid computing resources are typically managed by a local resource manager which implements allocation and prioritization policies while optimizing the execution of all submitted jobs for efficiency and performance according to site policy. GRAM is not a resource scheduler, but rather a protocol engine for communicating with a range of different local resource schedulers using a standard message format.

DRAFT

Table of Contents

DRAFT

List of Tables

DRAFT

Chapter 1. Conceptual details

A number of concepts underly the purpose and motivation for GRAM. These concepts are divided into broad categories below.

1. Targeted job types

GRAM is meant to address a range of jobs where arbitrary programs, reliable operation, stateful monitoring, credential management, and file staging are important. GRAM is not meant to serve as a "remote procedure call" (RPC) interface for applications not requiring many of these features. Furthermore, its interface model and implementation may be too costly for such uses. The GRAM5 service protocols and implementation will always involve multiple round-trips to support these advanced features that are not required for simple RPC applications.

2. Component architecture

Rather than consisting of a monolithic solution, GRAM is based on a component architecture at both the protocol and software implementation levels. This component approach serves as an ideal which shapes the implementation as well as the abstract design and features.

Service model	For GRAM5, the **globus-gatekeeper** daemon and GSI library are used for secure communications and service dispatch. The **globus-job-manager** daemon implements the job management and file transfer functionality.
Local Resource Manager Adapters	GRAM provides a scripted plug-in architecture to enable extension with adapters to control a variety of local resource systems.

DRAFT

3. Security

Secure operation	GRAM5 uses SSL-based protocols to establish identity or provide other security tokens needed to authorize GRAM5 service requests. Once authorized, each instance of the job service runs as a local POSIX user. GRAM5 restricts job monitoring and management operations to those who are authorized by the local site policy.
Local system protection domains	To protect users from each other, the GRAM5 job manager and the jobs it starts are executed in separate local security contexts. Additionally, GRAM mechanisms used to interact with the local resource are designed to minimize the privileges required and to minimize the risks of service malfunction or compromise.
Credential delegation and management	A client delegates some of its rights to the GRAM service in order to allow it to perform file transfers on behalf of the client and send state notifications to registered clients. Additionally, GRAM5 provides

	per-job credentials so that job instances may perform further authentication with other services.
Audit	GRAM uses a range of audit and logging techniques to record a history of job submissions and critical system operations. These records may be used to assist with accounting functions as well as to further mitigate risks from abuse or malfunction.

DRAFT

Chapter 2. GT 6.0 GRAM5 Approach

1. Introduction

The GRAM5 software implements a solution to the job-management problem described above. This solution is specific to operating systems following the POSIX programming and security model.

2. Component architecture approach

GRAM5's job management services interact with local resource managers (LRMs) and other service components of GT 6.0 in order to support job execution with coordinated file staging.

2.1. GRAM5 Architecture

The GRAM5 service architecture consists of several components which work together to authenticate users, manage jobs, interface with the LRM, and stage files. These components are described in the following table.

Table 2.1. GRAM5 Service Components

Gatekeeper	The globus-gatekeeper service provides a network interface to the GRAM5 system. It authenticates client identities and starts Job Manager processes using the local user account to which the client identity is mapped. Typically, one instance of the globus-gatekeeper process runs to accept network connections, and forks a new short-lived process to process each new conneciton.
Job Manager	The globus-job-manager daemon processes job requests and coordinates file transfers. There is one long-lived instance of this per user per LRM and one short-lived instance per job.
Scheduler Event Generator	The globus-scheduler-event-generator process parses LRM-specific data relating to job startup, execution, and termination into an LRM-independent data format. There is optionally one instance of this program per LRM.
LRM Adapter	The LRM adapter provides an interface between the GRAM5 system components and the LRM. It provides concrete implementations of the submit, cancel, and poll functionality for a particular system's LRM and to generate job status change events.

2.2. External Components used by GRAM5

2.2.1. Local resource manager

GRAM5 uses a local resource manager (LRM) to schedule and run jobs on a compute element. GRAM5 supports several common LRM systems (Condor, Torque, Oracle GridEngine) and can also be configured to manage jobs without an LRM.

DRAFT

GT 6.0 GSI C Security: Key Concepts

DRAFT

GT 6.0 GSI C Security: Key Concepts

Overview

GSI uses *public key* cryptography (also known as asymmetric cryptography) as the basis for its functionality. Many of the terms and concepts used in this description of GSI come from its use of public key cryptography.

For a good overview of GSI contained in the Web Services-based components of GT4, see Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective[1].

A reference for detailed information about public key cryptography is available in the book Handbook of Applied Cryptography [2], by A. Menezes, P. van Oorschot, and S. Vanstone, CRC Press, 1996. Chapter 8 [3] of this book deals exclusively with public key cryptography.

The primary motivations behind GSI are:

- The need for secure communication (authenticated and perhaps confidential) between elements of a computational Grid.
- The need to support security across organizational boundaries, thus prohibiting a centrally-managed security system.
- The need to support "single sign-on" for users of the Grid, including delegation of credentials for computations that involve multiple resources and/or sites.

DRAFT

[1] ../../GT4-GSI-Overview.pdf
[2] http://www.cacr.math.uwaterloo.ca/hac/
[3] http://www.cacr.math.uwaterloo.ca/hac/about/chap8.pdf

Table of Contents

DRAFT

List of Figures

DRAFT

Chapter 1. Conceptual Details

1. Public Key Cryptography

The most important thing to know about public key cryptography is that, unlike earlier cryptographic systems, it relies not on a single key (a password or a secret "code"), but on two keys. These keys are numbers that are mathematically related in such a way that if either key is used to encrypt a message, the other key must be used to decrypt it. Also important is the fact that it is next to impossible (with our current knowledge of mathematics and available computing power) to obtain the second key from the first one and/or any messages encoded with the first key.

By making one of the keys available publicly (a public key) and keeping the other key private (a *private key*), a person can prove that he or she holds the private key simply by encrypting a message. If the message can be decrypted using the public key, the person must have used the private key to encrypt the message.

Important: It is critical that private keys be kept private! Anyone who knows the private key can easily impersonate the owner.

2. Digital Signatures

Using public key cryptography, it is possible to digitally "sign" a piece of information. Signing information essentially means assuring a recipient of the information that the information hasn't been tampered with since it left your hands.

To sign a piece of information, first compute a mathematical hash of the information. (A hash is a condensed version of the information. The algorithm used to compute this hash must be known to the recipient of the information, but it isn't a secret.) Using your private key, encrypt the hash, and attach it to the message. Make sure that the recipient has your public key.

To verify that your signed message is authentic, the recipient of the message will compute the hash of the message using the same hashing algorithm you used, and will then decrypt the encrypted hash that you attached to the message. If the newly-computed hash and the decrypted hash match, then it proves that you signed the message and that the message has not been changed since you signed it.

3. Certificates

A central concept in GSI authentication is the *certificate*. Every user and service on the Grid is identified via a certificate, which contains information vital to identifying and authenticating the user or service.

A GSI certificate includes four primary pieces of information:

- A subject name, which identifies the person or object that the certificate represents.
- The public key belonging to the subject.
- The identity of a *Certificate Authority (CA)* that has signed the certificate to certify that the public key and the identity both belong to the subject.
- The digital signature of the named CA.

Note that a third party (a CA) is used to certify the link between the public key and the subject in the certificate. In order to trust the certificate and its contents, the CA's certificate must be trusted. The link between the CA and its certificate must be established via some non-cryptographic means, or else the system is not trustworthy.

DRAFT

GSI certificates are encoded in the X.509 certificate format, a standard data format for certificates established by the Internet Engineering Task Force (IETF). These certificates can be shared with other public key-based software, including commercial web browsers from Microsoft and Netscape.

4. Mutual Authentication

If two parties have certificates, and if both parties trust the CAs that signed each other's certificates, then the two parties can prove to each other that they are who they say they are. This is known as *mutual authentication*. GSI uses the Secure Sockets Layer (SSL) for its mutual authentication protocol, which is described below. (SSL is also known by a new, IETF standard name: Transport Layer Security, or TLS.)

Before mutual authentication can occur, the parties involved must first trust the CAs that signed each other's certificates. In practice, this means that they must have copies of the CAs' certificates--which contain the CAs' public keys--and that they must trust that these certificates really belong to the CAs.

To mutually authenticate, the first person (*A*) establishes a connection to the second person (*B*).

To start the authentication process, *A* gives *B* his certificate.

The certificate tells *B* who *A* is claiming to be (the identity), what *A*'s public key is, and what CA is being used to certify the certificate.

B will first make sure that the certificate is valid by checking the CA's digital signature to make sure that the CA actually signed the certificate and that the certificate hasn't been tampered with. (This is where *B* must trust the CA that signed *A*'s certificate.)

Once *B* has checked out *A*'s certificate, *B* must make sure that *A* really is the person identified in the certificate.

B generates a random message and sends it to *A*, asking *A* to encrypt it.

A encrypts the message using his private key, and sends it back to *B*.

B decrypts the message using *A*'s public key.

If this results in the original random message, then *B* knows that *A* is who he says he is.

Now that *B* trusts *A*'s identity, the same operation must happen in reverse.

B sends *A* her certificate, *A* validates the certificate and sends a challenge message to be encrypted.

B encrypts the message and sends it back to *A*, and *A* decrypts it and compares it with the original.

If it matches, then *A* knows that *B* is who she says she is.

At this point, *A* and *B* have established a connection to each other and are certain that they know each others' identities.

5. Confidential Communication

By default, GSI does not establish confidential (encrypted) communication between parties. Once mutual authentication is performed, GSI gets out of the way so that communication can occur without the overhead of constant encryption and decryption.

GSI can easily be used to establish a shared key for encryption if confidential communication is desired. Recently relaxed United States export laws now allow us to include encrypted communication as a standard optional feature of GSI.

DRAFT

A related security feature is communication integrity. Integrity means that an eavesdropper may be able to read communication between two parties but is not able to modify the communication in any way. GSI provides communication integrity by default. (It can be turned off if desired). Communication integrity introduces some overhead in communication, but not as large an overhead as encryption.

6. Securing Private Keys

The core GSI software provided by the Globus Toolkit expects the user's private key to be stored in a file in the local computer's storage. To prevent other users of the computer from stealing the private key, the file that contains the key is encrypted via a password (also known as a passphrase). To use GSI, the user must enter the passphrase required to decrypt the file containing their private key.

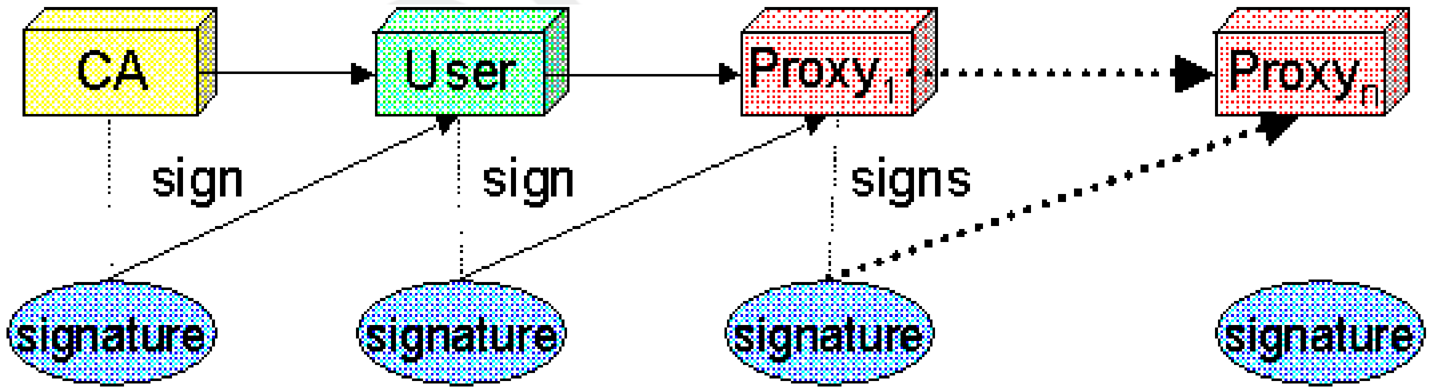
We have also prototyped the use of cryptographic smartcards in conjunction with GSI. This allows users to store their private key on a smartcard rather than in a file system, making it still more difficult for others to gain access to the key.

7. Delegation, Single Sign-On and Proxy Certificates

GSI provides a delegation capability: an extension of the standard SSL protocol which reduces the number of times the user must enter his passphrase. If a Grid computation requires that several Grid resources be used (each requiring mutual authentication), or if there is a need to have agents (local or remote) requesting services on behalf of a user, the need to re-enter the user's passphrase can be avoided by creating a *proxy*.

A proxy consists of a new certificate and a private key. The key pair that is used for the proxy, i.e. the public key embedded in the certificate and the private key, may either be regenerated for each proxy or obtained by other means. The new certificate contains the owner's identity, modified slightly to indicate that it is a proxy. The new certificate is signed by the owner, rather than a CA. (See diagram below.) The certificate also includes a time notation after which the proxy should no longer be accepted by others. Proxies have limited lifetimes.

Figure 1.1. The new certificate is signed by the owner, rather than a CA.



The proxy's private key must be kept secure, but because the proxy isn't valid for very long, it doesn't have to kept quite as secure as the owner's private key. It is thus possible to store the proxy's private key in a local storage system without being encrypted, as long as the permissions on the file prevent anyone else from looking at them easily. Once a proxy is created and stored, the user can use the *proxy certificate* and private key for mutual authentication without entering a password.

When proxies are used, the mutual authentication process differs slightly. The remote party receives not only the proxy's certificate (signed by the owner), but also the owner's certificate. During mutual authentication, the owner's public key (obtained from her certificate) is used to validate the signature on the proxy certificate. The CA's public key is then used to validate the signature on the owner's certificate. This establishes a chain of trust from the CA to the proxy through the owner.

Note

GSI, and software based on it (notably the Globus Toolkit, GSI-SSH, and GridFTP), is currently the only software which supports the delegation extensions to TLS (a.k.a. SSL). The Globus Alliance has worked in the GGF and the IETF to standardize this extension in the form of Proxy Certificates (RFC 3820) [http://www.ietf.org/rfc/rfc3820.txt].

DRAFT

Chapter 2. Related Documents

- Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective[1]
- Handbook of Applied Cryptography [2]

DRAFT

Glosary

DRAFT

Glosary

DRAFT

Table of Contents

DRAFT

Glossary

C

Certificate Authority (CA)	An entity that issues certificates.

G

Grid Resource Allocation and Management (GRAM)	This component is used to locate, submit, monitor, and cancel jobs on Grid computing resources.
GridFTP	A file transfer protocol based on FTP with extensions for security and parallel data transfers.
Grid Security Infrastructure (GSI)	GSI stands for Grid Security Infrastructure and is used to describe the original infrastructure of GT security, which is comprised of SSL, PKI and proxy certificates.

L

Local Resource Manager (LRM)	A system which controls access to a compute resource, such as a compute cluster or parallel computer. Such systems provide batch execution interfaces, which GRAM uses to execute jobs. *Condor*, *Portable Batch System*, *GridEngine* are examples of local resource managers. See Also Condor, Portable Batch System, Oracle GridEngine.
LRM Adapter	The interface code between a *Local Resource Manager* and GRAM. In most cases, this consists of a Perl module that implements the `Globus::GRAM::JobManager` class and a *Scheduler Event Generator* module. See Also Local Resource Manager.

P

private key	The private part of a key pair. Depending on the type of certificate the key corresponds to it may typically be found in `$HOME/.globus/userkey.pem` (for user certificates), `/etc/grid-security/hostkey.pem` (for host certificates) or `/etc/grid-security/<service>/<service>key.pem` (for service certificates). For more information on possible private key locations see this.
proxy certificate	A short lived certificate issued using a EEC. A proxy certificate typically has the same effective subject as the EEC that issued it and can thus be used in its place. GSI uses proxy certificates for single sign on and delegation of rights to other entities. For more information about types of proxy certificates and their compatibility in different versions of GT, see http://dev.globus.org/wiki/Security/ProxyCertTypes.

DRAFT

public key	The public part of a key pair used for cryptographic operations (e.g. signing, encrypting).

DRAFT