Acceso Seguro a Datos Confidenciales en *Grids*

Yudith Cardinale, Carlos Figueira y Emilio Hernández*
Universidad Simón Bolívar,
Departamento de Computación y Tecnología de la Información,
Caracas 1080-A, Venezuela
{yudith,figueira,emilio}@usb.ve

Resumen

La tecnología *grid* permite compartir los recursos de almacenamiento y de cómputo de distintas organizaciones geográficamente distribuidas. Se establece un control de acceso a estos recursos a través de una plataforma de seguridad única basada, por ejemplo, en infraestructuras de Clave Pública y Organizaciones Virtuales. Existen escenarios en los que las fuentes de datos no son incorporadas al *grid*, o que los datos son confidenciales, pero para su procesamiento deben ser accedidos desde recursos computacionales del *grid*. Un ejemplo de esto es el caso de investigaciones sobre imágenes médicas, donde se necesita mucho poder de cómputo para el análisis de las imágenes, pero el acceso a los datos requiere un control muy estricto, tanto por razones éticas como legales. En estos casos, la clave es permitir el acceso a las fuentes de datos bajo control de usuarios autorizados. En este trabajo se describen varios escenarios de acceso a datos confidenciales en repositorios externos al *grid*. Se evalúa la capacidad de operación en estos contextos para dos tipos de plataformas *grid*: plataformas basadas en Globus y SUMA/G. Se presenta un esquema para el acceso seguro bajo el control del usuario, y se proponen mecanismos en ambas plataformas para el correcto manejo de todos los escenarios descritos.

Palabras claves: Grids, Grids de datos, Globus Toolkit, Seguridad, Certificados digitales, Fuentes de datos, Datos Confidenciales.

*Este trabajo recibió apoyo financiero del Fonacit (*Proyecto Agenda Petróleo* 97003588) y de la Universidad Simón Bolívar (*Programa de Apoyo a Grupos de Investigación, Decanato de Investigación y Desarrollo*).

1. Introducción

La tecnología *grid* [6] permite compartir los recursos de almacenamiento y facilidades de cómputo de distintas organizaciones geográficamente distribuidas. Involucra una racionalización del uso de estos recursos, por cuanto permite a cualquier institución utilizar las capacidades computacionales ociosas de otra y, lo que es más importante, genera una cultura cooperativa en el uso de la información y el conocimiento que se produce en las distintas instituciones. Un *grid* ofrece transparencia en el acceso a recursos remotos, manejo de recursos de manera segura bajo una plataforma de seguridad global y control de acceso a través de *Organizaciones Virtuales*.

El problema general de acceso a fuentes de datos en un *grid* se enfoca en niveles, relacionados con la transferencia propiamente dicha [1], el manejo de réplicas para garantizar alta disponibilidad y alternativas de acceso para incrementar la eficiencia en el acceso [5] y la independencia de la localización, a través de directorios que establecen la correspondencia de nombres globales a direcciones específicas o URLs (GUID: *Globally Unique Identifier*). Aspectos de más alto nivel, relacionados con la organización de los contenidos, metadatos y ontologías, entre otros, se dejan a las capas de aplicación dentro del *grid*.

Dentro de un *grid*, el acceso a cualquier recurso, y por ende a los repositorios de datos, está típicamente protegido con un mecanismo de certificados digitales. Debido al carácter multi-institucional de un *grid*, este nivel básico de protección puede no ser suficiente en algunos escenarios, por ejemplo, vinculados con la participación de organizaciones que necesitan que sus datos sean procesados en el *grid*, pero asegurando su confidencialidad. Algunos escenarios requieren de un alto nivel de confidencialidad, ya sea por razones comerciales, de violación de datos individuales privados, o de seguridad nacional, entre otras. Por ejemplo, los datos pueden tener requerimientos de privacidad, como los expedientes médicos o datos asociados que pueden ser utilizados con propósitos clínicos y de investigación. En este caso el requerimiento de privacidad no sólo está relacionado con el factor de confianza, sino que tiene además un componente legal. Por supuesto, una opción para una institución con estos requerimientos puede ser no formar parte de una plataforma *grid*, pero esta alternativa le privaría de las ventajas de utilizar valiosos recursos de cómputo disponibles en otras instituciones. Dado que los datos pueden ser procesados en plataformas diferentes a las que proveen los datos, puede ser necesaria la transferencia de archivos a equipos para cómputo, abriendo una brecha de posible acceso incontrolado a datos confidenciales.

El problema de acceso seguro remoto a datos confidenciales ha sido tratado en diferentes contextos. En [12, 13] se presentó un sistema de archivo para acceso remoto seguro, llamado *Self-certifying secure file system*, SFS. Se trata de una extensión del sistema de archivos, ideado como un reemplazo seguro de Network File System (NFS), que permite el acceso a archivos de manera distribuida, accediendo a servidores que pueden estar distribuidos en Internet. Los servidores son autenticados a través de su clave pública, el cual se incluye en el nombre de archivo; ofrece al usuario el control sobre la autenticación para el acceso a sus datos, y establece un canal cifrado para la transferencia de archivos. Este esquema es apropiado para aplicaciones que corren bajo el control de un usuario en máquinas específicas, pero no se adaptan bien al entorno *grid*, en el cual la ejecución de la aplicación se delega en componentes del *middleware*. En [9] se propone el uso de una plataforma GSI-SFS, que combina las capacidades de SFS (acceso a datos controlado por el usuario) con la plataforma de seguridad de Globus GSI [7, 2] (autenticación "una sola vez", delegación, manejo unificado de claves). En [14] se describe un esquema de autenticación y acceso controlado a recursos (*Sygn*) sobre una plataforma experimental de *grid* (μ*Grid*) orientada a aplicaciones médicas. Debido a lo sensible de los datos utilizados por estas aplicaciones, se implementa un control fino de acceso a datos manejado por usuarios autorizados.

Sin embargo, estas plataformas sólo consideran los escenarios en los que las fuentes de datos pertenecen al *grid*. En casos de datos sensibles (por ejemplo, imágenes médicas), los administradores de las fuentes de datos no aceptarán delegar el control del acceso a sus recursos a terceras partes, incluyendo el *grid*.

En este trabajo nos concentramos en el aspecto de ofrecer mecanismos y estrategias para ayudar a mantener la privacidad del acceso a los datos. Principalmente se cubren dos condiciones de contexto que son (i) la posibilidad de copiar archivos confidenciales desde el repositorio de datos a la plataforma de cómputo y (ii) el nivel de confianza que se tenga en los mecanismos del *grid* para proveer acceso a los repositorios de datos. Se evalúan los escenarios generados por las diferentes combinaciones de las condiciones de contexto mencionadas. Se propone solución a los problemas de acceso a datos planteados por los diferentes escenarios en el contexto de plataformas *grid* basadas en Globus, y SUMA/G, una plataforma *grid* basada en CORBA y el Globus Toolkit [15].

En la sección 2 se presenta una introducción a las plataformas *grid*, en particular, la plataforma basada

en Globus LCG [11], y SUMA/G [3]. La sección 3 describe en detalle los escenarios de acceso a datos remotos seguros desde el *grid*; para cada caso, se presentan los esquemas en LCG y SUMA/G que resuelven el problema. Por último, se presentan algunas conclusiones del trabajo en la sección 4.

2. Plataformas grid

Las plataformas *grid* permiten la ejecución de trabajos sobre recursos remotos de manera sencilla y transparente. Un *grid* incluye una serie de componentes para implementar sus servicios. Algunos de los más importantes son:

- Un punto de entrada al *grid*, que permite a los usuarios acceder a servicios tales como envío de trabajos, acceso y envío de datos, y consulta de estado de recursos.
- Una plataforma de seguridad, que incluye servicios de control de acceso a recursos, autenticación de usuarios (típicamente "una sola vez", *single sign on*) y privacidad.
- Recursos, tanto de ejecución de tareas como de almacenamiento de datos.
- Administrador de recursos que se encarga de llevar el registro de los recursos del *grid*, y de seleccionar los más convenientes para satisfacer las solicitudes de los usuarios.

En esta sección se resumen las características más relevantes de las plataformas *grid* basadas en Globus (en particular, LCG); y la plataforma SUMA/G, la cual está orientada a la ejecución de aplicaciones Java, y que presenta valor agregado en términos de transparencia.

2.1. Plataformas basadas en Globus

El Globus Toolkit (GT) ofrece servicios que pueden ser combinados para desarrollar infraestructuras de *grid*. Éstos cubren servicios de administración y supervisión de recursos, seguridad, administración de archivos, comunicación y ejecución de trabajos. LCG utiliza los servicios distribuidos en la versión 2 del Globus Toolkit (GT2).

2.1.1. Arquitectura de LCG

Seguridad (GSI) La plataforma de seguridad se basa en la Infraestructura de Clave Pública (PKI) y el concepto de Organizaciones Virtuales (VO). Las VO identifican grupos de usuarios del *grid* en (posiblemente) diferentes instituciones que comparten los mismos derechos de acceso a recursos. GSI permite la autenticación segura y comunicación sobre una red abierta. Usa cifrado de clave pública, certificados X.509 y el protocolo de comunicación SSL. Estos estándares se han extendido para proveer delegación y autenticación de "una sola vez" (*single sign on*)

Interfaz de usuario (UI) Designa cada una de las máquinas que proveen punto de acceso al *grid*. Los usuarios instalan su certificado en su cuenta en una o más de estas máquinas, desde las cuales envían trabajos y acceden al resto de los servicios de *grid*.

Elemento de Cómputo (Computing Element, CE) Un CE recibe los trabajos para ejecución sobre sus nodos de ejecución asociados (*Worker Nodes*, WN). Si la función del nodo es servir de almacenamiento de datos, se denomina Elemento de Almacenamiento (*Storage Element*, SE). Los SEs proveen acceso uniforme a los recursos de almacenamiento.

Servicio de Información (IS) Provee la información sobre los recursos del *grid* y su estado. Con base en esta información, otros componentes del *grid* pueden administrar dichos recursos, asignar trabajos, etc.

Manejo de trabajos Los servicios del Sistema de Manejo de Carga de Trabajo (WMS) se encargan de aceptar los trabajos enviados al *grid* y de reenviar dichos trabajos al CE apropiado, según la disponibilidad de recursos y requerimientos del trabajo.

2.1.2. *Ejecución de aplicaciones en* LCG

Los pasos de ejecución de trabajos en LCG, característicos de los *grids* basado en Globus, se resumen a continuación. Antes de usar los servicios del *grid*, el usuario debe obtener un certificado digital, estar registrado en al menos una VO y obtener una cuenta en un UI.

1. El usuario se conecta a la UI y crea un certificado temporal (*proxy* [2]), que se usa para la sesión y tiene duración limitada. Este certificado será utilizado para autenticar al usuario en las interacciones entre todos los componentes del *grid* involucrados en los servicios requeridos por el usuario.

2. El usuario crea una descripción de su trabajo (JDL), en la cual puede especificar, entre otros atributos del trabajo, los archivos que serán copiados del UI al CE seleccionado (*Input Sandbox*) y viceversa (*Output Sandbox*).

3. Los componentes del núcleo del *grid* cooperan para seleccionar un CE para ejecutar el trabajo, y le envían los archivos descritos en el JDL (por ejemplo, ejecutables y archivos de entrada).

4. Durante la ejecución, el trabajo puede acceder a los archivos enviados al CE, o a otros archivos residentes en algún SE.

5. Cuando el trabajo finaliza, el usuario puede obtener los resultados a través de la UI.

2.2. SUMA/G

SUMA/G (Scientific Ubiquitous Metacomputing Architecture/Globus) es una plataforma *grid* para la ejecución transparente de *bytecode* de Java en máquinas remotas. SUMA/G extiende el modelo de ejecución de Java a ambientes *grid*; en particular, las clases y los datos son cargados dinámicamente durante la ejecución.

El *middleware* de SUMA/G fue originalmente construido usando software y tecnologías de comunicación de libre acceso, incluyendo Java y CORBA [4]. Más recientemente se le han incorporado gradualmente servicios generales de Globus usando el Java Cog Kit [16]. Esto último le permite a un *grid* SUMA/G conectarse o integrarse a otros *grids* basados en Globus, así como beneficiarse de la evolución de los servicios y de las herramientas de Globus. La Figura 1 muestra la arquitectura general de SUMA/G, cuyos componentes son descritos en la sección 2.2.2.

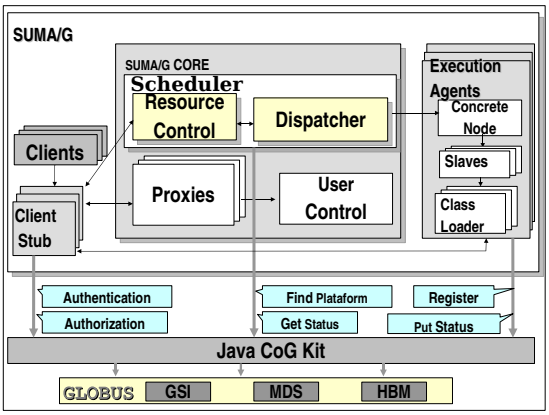

Figura 1: Arquitectura de SUMA/G

2.2.1. *Ejecución de aplicaciones en* SUMA/G

El modelo básico para la ejecución de programas Java en SUMA/G es muy simple.

1. Los usuarios inician la ejecución de sus programas a través de comandos sencillos ejecutados en la máquina cliente. Se puede invocar tanto `sumag Execute`, correspondiente al modo de ejecución en línea, o `sumag Submit`, que permite ejecución fuera de línea (ejecución en *batch*).

2. Al momento de la invocación del servicio de ejecución, se genera un certificado *proxy* (usando GSI) que representa al usuario y permite que los procesos creados a partir de este momento puedan adquirir recursos del *grid* sin necesidad de intervención adicional del usuario.

3. Una vez que el núcleo de SUMA/G recibe el requerimiento de la máquina cliente, autentica al usuario (a través de GSI), de manera transparente selecciona una plataforma de ejecución (apoyándose en los servicios de MDS), y re-envía el mensaje de requerimiento a dicha plataforma.

4. El Agente de Ejecución, que se ejecuta en la plataforma designada, recibe un objeto que representa la aplicación e inicia, en una Máquina Virtual Java independiente, un Agente de Ejecución Esclavo, quien será el que finalmente ejecute la aplicación. El `Class Loader` de SUMA/G se inicia en la nueva máquina virtual; su función es cargar las clases y los datos durante la ejecución. Las posibles fuentes de clases y datos de entrada, así como los posibles destinos para las salidas generadas durante la ejecución, incluyen:

   a) la máquina (cliente) donde se invocó el servicio de ejecución de la aplicación

   b) sistemas de archivos remotos en los que el usuario tiene una cuenta. El acceso a estos archivos desde SUMA/G es manejado por el Servidor de Cuentas Remotas

   Un esquema de componentes intercambiables (*plug-ins*) permite la implementación de diversos protocolos para administrar las clases y los archivos (como CORBA, sftp, gridFTP).

En el caso de invocar el servicio `sumag Execute`, el usuario sólo debe especificar el nombre de la clase principal. El resto de las clases y archivos de datos son cargados por demanda en tiempo de ejecución. La entrada y salida estándar son manejados transparentemente, dando la ilusión de una ejecución local.

Para el caso del servicio `sumag Submit`, el cliente empaqueta de manera transparente las clases y archivos de entrada y las entrega al núcleo de SUMA/G; la salida es mantenida en SUMA/G hasta que el usuario la solicite. Algunos otros atributos de ejecución pueden ser opcionalmente indicados al momento de solicitar los servicios desde la máquina cliente, por ejemplo, especificaciones de planificación o de plataformas.

2.2.2. Componentes de SUMA/G

A continuación se describen los componentes de SUMA/G y su rol en la ejecución de las aplicaciones:

Intermediario (*Proxy*) Recibe un objeto del Cliente (*Client Stub*) que contiene los datos que describen la aplicación, tales como el nombre de la clase principal, sus argumentos, especificaciones de planificación (opcional) y estructuras de datos que reducen los tiempos de comunicación (llamados *datos pre-cargados*, también opcionales). Luego de verificar los permisos del usuario, el Intermediario solicita al Planificador una plataforma de ejecución adecuada y envía el objeto que describe la aplicación a la plataforma seleccionada. Para el caso de ejecución de trabajos fuera de línea o *batch*, el Intermediario mantiene los resultados hasta que el usuario los solicite.

Planificador (*Scheduler*) Responde las solicitudes del Intermediario basándose en los requerimientos de la aplicación y la información de estado global de la plataforma *grid*. El algoritmo de planificación trata de mantener balance de carga en el *grid*. Por otro lado, el Planificador también es usado para registrar los recursos de SUMA/G tales como las plataformas de ejecución o los conjuntos de datos disponibles en localizaciones específicas. Además mantiene la información de la plataforma *grid* tanto estática como dinámica, tales como tamaño de la memoria, librerías disponibles y carga promedio. El servicio MDS de Globus es usado por SUMA/G para el registro de los componentes; de esta manera, el Planificador de SUMA/G interroga al MDS para encontrar una plataforma adecuada cuando recibe un requerimiento para la ejecución de un trabajo particular.

Control de Usuario (*User Control*) Es el encargado de la administración de los usuarios. El GSI es usado para la autorización y autenticación en SUMA/G, y constituye el mecanismo para incluir todos los componentes de SUMA/G en el espacio de seguridad del *grid*.

Cliente (*Client Stub*) Crea el objeto que representa la aplicación, extrae del *grid* los resultados y la información de desempeño posterior a la ejecución y atiende los requerimientos del Agente de Ejecución (a través de *callbacks*) para cargar las clases y datos dinámicamente. Este componente se ejecuta en la máquina del usuario o en una máquina servidora o portal de SUMA/G. En cualquiera de los dos casos, el usuario debe tener un certificado válido instalado en dicha máquina.

Agente de Ejecución (*Execution Agent*) Cuando se inicia un Agente de Ejecución, se registra con el Planificador como un nuevo recurso disponible. Durante las operaciones de ejecución de aplicaciones, recibe del Intermediario el objeto que representa la aplicación e inicia la ejecución, cargando clases y datos dinámicamente desde el cliente o desde sistemas de archivos remotos a través del SUMA/G Class Loader y del subsistema de I/O de SUMA/G. Una vez que la aplicación finaliza, el Agente de Ejecución retorna los resultados al cliente. Para plataformas paralelas, el Agente de Ejecución juega el papel de *front-end*. Sólo el *front-end* de la plataforma paralela se registra en SUMA/G, ya sea como una plataforma mpiJava, o como una granja de recursos de cómputo para múltiples ejecuciones de trabajos independientes. En el caso de plataformas de un único nodo, aunque existe un único Agente de Ejecución por plataforma, éste puede ejecutar varias aplicaciones concurrentemente, delegando un Esclavo diferente para cada proceso del usuario. En el caso de plataformas de múltiples nodos (o plataformas paralelas), el componente Agente de Ejecución espera en el front-end por requerimientos de ejecución y, al recibir uno, inicia un Esclavo para cada aplicación paralela en un nodo diferente.

Subsistema de Entrada/Salida Considera cuatro aspectos:

1. la carga dinámica de clases. El hecho de instanciar un SUMA/G `Class Loader` independiente para cada aplicación a ejecutar, permite la carga dinámica de las clases referenciadas por la clase principal de la aplicación a ejecutar, desde el cliente hasta el `Agente de Ejecución` a tiempo de ejecución
2. el redireccionamiento de la entrada, salida y error estándar. Cada vez que se hace una petición a un `Agente de Ejecución` se modifica el ambiente de ejecución para que la salida, entrada y error estándar estén conectados al cliente que originó la petición, dando la ilusión de una ejecución local (para ejecuciones interactivas)
3. el redireccionamieno del `java.io`. El `suma.io` es un paquete de redefine las clases básicas de `java.io`. De esta manera, por medio de *callbacks*, se acceden los datos en las fuentes de datos especificadas y no en el Agente de Ejecución
4. el almacenaje intermedio de datos (*buffering*). La carga remota de archivos desde las fuentes especificadas hacia el `Agente de Ejecución` se realiza utilizando un sistema de *buffering*, que reduce significativamente el número de *callbacks* de lectura y escritura al cliente. Esto permite leer (operación de lectura solicitada por la aplicación en ejecución) una mayor cantidad de información, adelantándose y abasteciendo el requerimiento de lectura real de la aplicación. Igualmente, la escritura remota se efectúa cuando se llena el *buffer* de escritura o se cierra el archivo. A través de esta técnica se reduce el consumo de tiempo adicional (*overhead*) generado por el acceso remoto. El tamaño de los *buffers* de lectura y escritura son parametrizables por el usuario, adaptándolo a los requerimientos de su aplicación.

La Figura 2 muestra el esquema de Entrada y Salida de SUMA/G.

3. Privacidad en acceso a datos externos al grid

Si un repositorio utiliza certificados aceptables por el *grid*, y confía en la administración de las Organizaciones Virtuales de éste (en particular, en su Autoridad Certificadora) entonces las solicitudes de acceso a datos desde el *grid* pueden ser autorizadas. En este caso, la solicitud vendrá acompañada de un certificado válido, el cual es verificado y admitido por el repositorio. Hablamos de *confianza mutua* en este escenario. Sin embargo, también es posible el escenario en que una organización pertenece al *grid* pero conserva sus propios mecanismos de acceso, independientes de los mecanismos de acceso estándares del *grid*. En caso de que no utilice los certificados admitidos por el *grid*, debe proveerse algún mecanismo, controlado directamente por los usuarios autorizados, para que sus aplicaciones accedan a los datos administrados por esta organización.

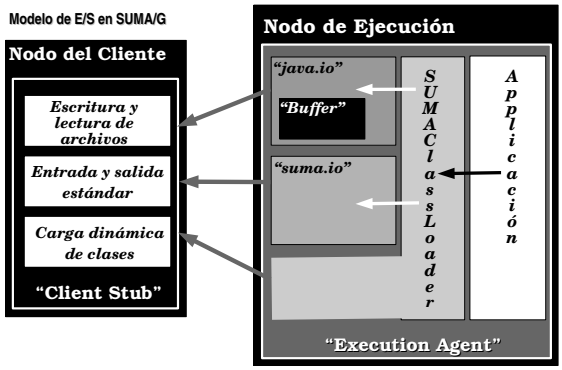

Figura 2: Subsistema de entrada y salida de SUMA/G

Este mecanismo establece una correspondencia entre certificados del *grid* y la autenticación (de acuerdo al mecanismo usado en el repositorio) del usuario en el repositorio. Este tipo de correspondencia entre dominios de autenticación es utilizado por GSI-SFS [9]. En este caso hablamos de un escenario *sin confianza mutua*, aun en los casos en que no sea estrictamente un problema de confianza sino un problema de restricción legal.

Por otro lado, en el contexto de las plataformas *grid*, los repositorios de datos son parte del *grid*, o los datos son suministrados por el usuario junto con la tarea a ejecutar. Estos datos son procesados por dicha tarea, y los resultados son, igualmente, enviados al usuario (es decir, a la máquina que sirve al usuario como punto de acceso al *grid*) o almacenados en los repositorios del *grid*. Los datos en principio sólo son accesibles por los usuarios autorizados, pero al poderse copiar los datos a las plataformas de cómputo, decimos que *las copias de archivos están autorizadas*. También consideramos escenarios en los que los datos no pueden ser copiados a medios físicos persistentes de otras organizaciones del *grid*, por razones de confidencialidad. El usuario tiene acceso tanto al *grid* como al repositorio de datos; requiere de los recursos de cómputo del *grid* para procesarlos pero no desea o no le está permitido que los datos sean almacenados fuera del repositorio original. Por lo tanto, el programa debe hacer accesos remotos a los datos durante la ejecución sin copiar los archivos en discos locales, es decir, *las copias de archivos no están autorizadas*.

En la sección 3.1 presentamos un esquema para el acceso desde el *grid* en escenarios sin confianza mutua. La sección 3.2 describe las diferentes combinaciones generadas por estos casos; se provee para cada caso un esquema de solución en SUMA/G y se compara con plataformas de *grid* basadas en Globus.

3.1. Esquema para acceso seguro en escenarios sin confianza mutua

En estos escenarios, los esquemas de autenticación y control de acceso usados por el *grid* no son admitidos por los administradores del repositorio de datos externo al *grid*. Para que las aplicaciones de los usuarios con acceso a estos datos puedan ejecutarse en el *grid*, debe establecerse un mecanismo que permita a las aplicaciones autenticarse ante los repositorios en nombre del usuario.

El esquema propuesto se basa en el establecimiento de una *clave de autenticación y cifrado de sesión temporal* (que llamaremos CAEST), similar a las certificados temporales (*proxy*) de Globus, y la infraestructura de PKI. La CAEST contiene los siguientes elementos:

- *ID*, identificación del usuario ante el repositorio, que depende del mecanismo de autenticación usado por dicho repositorio (por ejemplo, su *login*)
- *AT*, registro de autenticación temporal. Por ejemplo un *OTP* (*one-time password OTP* [8]) o *magic cookie* [10]), que se usa una sola vez y tiene duración limitada.
- El *hash* criptográfico (por ejemplo, MD5 o SHA-1) de la concatenación del *ID* y el *AT*.

Asociado al repositorio de datos, se instala un servidor que cumple dos funciones: atender las solicitudes de CAEST de usuarios autorizados, y controlar los accesos a datos a través de las CAEST presentadas por las aplicaciones. Este servidor, así como los usuarios, poseen su par de claves pública/privada.

Generación de la CAEST El usuario solicita una CAEST a través de una conexión segura (usando ssh, por ejemplo). La solicitud del usuario incluye un *cookie* cifrado con la clave pública del servidor. El servidor: i) crea un *AT*, que junto con el *ID* del usuario, se cifra con la clave pública del servidor; ii) calcula el *hash(ID,AT)*, y lo cifra con la clave secreta del servidor de cuentas; iii) instala el registro creado en su base de datos de control de acceso; iv) descifra el *cookie* del usuario con su clave privada; v) cifra la CAEST y el *cookie* del usuario con la clave pública del usuario, y la envía al usuario.

El usuario descifra la CAEST y el *cookie* con su clave secreta. El *cookie* permite verificar que la respuesta corresponde a su solicitud, y que fue efectivamente realizada por el servidor.

Acceso al repositorio usando CAEST El usuario debe entregar la CAEST al servidor que ejecutará su aplicación en el *grid*, a través de un canal seguro, de manera de poder usarla en el acceso a datos. En la sección 3.2.2 y 3.2.4 describimos los detalles de cómo se usa el mecanismo en el contexto de SUMA/G. El servidor de cuentas autorizará el acceso a los archivos de la cuenta del usuario al verificar la CAEST, de acuerdo a los siguientes pasos:

1. Descifrar el *ID* y el *AT* con su clave secreta
2. Descifrar con su clave pública el *hash* incluido en el CAEST
3. Calcular el *hash* de (*ID,AT*) y verificar que coincida con el de la CAEST
4. Verificar que el *AT* corresponde con el generado para el usuario *ID* y que es válido. Para que sea válido, no puede haber sido usado antes, y no debe haber expirado.

La figura 3 muestra el esquema de generación de CAEST y acceso a un repositorio usando dicha CAEST.

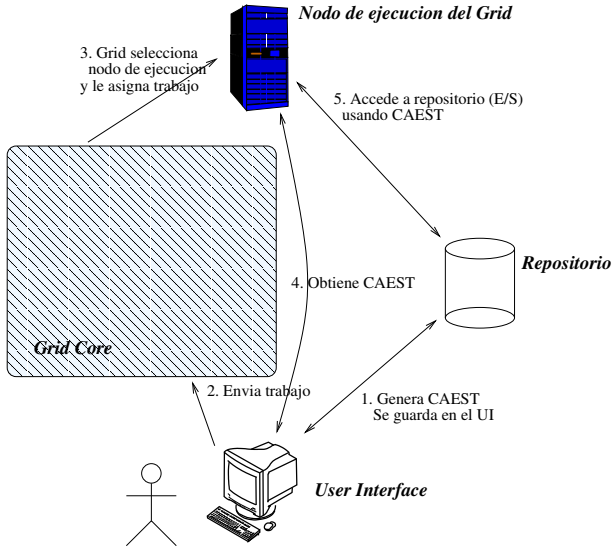

Figura 3: Acceso a repositorio usando una clave de autenticación y cifrado temporal (CAEST)

3.2. Descripción de escenarios y soluciones

3.2.1. Copias de archivo autorizadas, confianza mutua

Este es el caso trivial. La confianza mutua permite que se autoricen los accesos al repositorio externo de datos desde el *grid*. La autorización de copia de archivos permite que los datos puedan ser copiados al *grid* para su procesamiento.

Plataformas de grid basadas en Globus.

En plataformas de *grid* basadas en Globus, se podrían utilizar dos posibles mecanismos básicos:

1. El usuario copia los datos del repositorio al *grid*, y posteriormente envía la solicitud de ejecución. Si los datos son de tamaño pequeño, pueden enviarse junto con la solicitud. En caso contrario, deben copiarse a un *Storage Element* del *grid* y hacer la referencia apropiadamente en la aplicación.

2. Establecer una serie de componentes adicionales al *grid* que permitan traducir las referencias a archivos remotos con los certificados del *grid* en órdenes de copia de los archivos al *grid* con los mecanismos de autenticación locales (por ejemplo, identificación de usuario y clave del usuario enviados a través de un canal seguro entre el *grid* y el repositorio).

Solución en SUMA/G.

El servidor de cuentas remotas de SUMA/G, que se ejecuta en cada repositorio de datos en el espacio del usuario, permite servir los accesos remotos controlados a los datos. El origen y destino de los archivos para la aplicación se especifica como un atributo de la ejecución (un URL). Esta información es utilizada por el Agente de Ejecución de SUMA/G para redireccionar los accesos (lecturas y escrituras) a archivos de la aplicación en la cuenta remota especificada por el URL. Tanto las lecturas como las escrituras van al mismo sitio (directorio en repositorio indicado como atributo de la ejecución). La autenticación se verifica entre la aplicación y el servidor de cuentas remotas usando certificados; la comunicación entre ellos va cifrada a través de TLS/SSL.

3.2.2. Copias de archivo autorizadas, sin confianza mutua

Si no hay confianza mutua entre el *grid* y el repositorio de datos, el mecanismo mencionado en la sección 3.2.1, que hace las veces de puente entre los dos dominios de seguridad, no es posible. Por lo tanto, los mecanismos utilizados deben estar bajo control del usuario, con la restricción de que no puede ceder al *grid* su clave de acceso (o certificado). La copia de datos, aunque posible, exige que se resuelva el problema de la autenticación para autorizar el acceso.

Plataformas de grid basadas en Globus.

Cuando los datos están fuera del *grid*, el usuario debe encargarse de obtener los archivos de entrada y subirlos al *grid*, así como recuperar los archivos de salida del *grid* y enviarlos al repositorio luego de ejecutar la aplicación (si es el caso). Este proceso puede ser simplificado a través del desarrollo de interfaces de programación que admitan hacer referencias a recursos fuera del *grid* combinándolo con un esquema de CAEST, o utilizando esquemas como GSI-SFS.

Solución en SUMA/G.

La combinación del mecanismo inherente en SUMA/G para acceso a archivos remotos, con el mecanismo para acceso seguro bajo control del usuario descrito antes (sección 3.1), permite resolver los problemas que plantea este escenario: i) el usuario solicita la ejecución en el *grid* de una aplicación, especificando la opción de seguridad en acceso remoto; ii) cuando la aplicación es asignada a un nodo del *grid*, el correspondiente Agente de Ejecución se comunica con el Cliente para obtener la CAEST; iii) durante la ejecución, cuando se captura una referencia a archivo remoto, el Agente de Ejecución contacta al servidor en el repositorio a través de una conexión TLS/SSL, y presenta la CAEST para esa ejecución; iv) una vez autorizada, la aplicación accederá a los datos en el repositorio externo como si estuviera en servidores pertenecientes al *grid*.

3.2.3. Copias de archivo no autorizadas, confianza mutua

Si las copias de archivo no son autorizadas, debe evitarse que sean guardadas en repositorios o nodos del *grid*. Adicionalmente, deben enviarse cifrados.

Plataformas de grid basadas en Globus.

El esquema donde los archivos no pueden ser copiados al *grid* no está contemplado.

Solución en SUMA/G.

SUMA/G se adapta naturalmente a este contexto. El manejo de archivos de SUMA/G se basa en que los archivos son leídos (y escritos) remotamente sin necesidad de copiarlos en discos locales mediante el uso de *buffers*. El sistema de *buffering* de SUMA/G no sólo permite mantener los datos en memoria sino además permite lecturas adelantadas y agrupamiento de escrituras, reduciendo así las comunicaciones con el repositorio remoto.

3.2.4. Copias de archivo no autorizadas, sin confianza mutua

Plataformas de grid basadas en Globus.

El esquema donde los archivos no pueden ser copiados al *grid* no está contemplado. Incluso usando mecanismos como GSI-SFS no podemos controlar que no queden copias en los nodos de ejecución. Por lo tanto, se requiere de nuevos esquemas para operar en estos escenarios.

Solución en SUMA/G.

Al igual que en el escenario con copias autorizadas y sin confianza mutua (sección 3.2.2), la combinación del modelo inherente de manejo de archivos en SUMA/G (sistema de *buffering*) junto con la provisión de soporte de CAEST permite que SUMA/G se adapte bien a este contexto.

4. Conclusiones

El *grid* ofrece una plataforma atractiva para aprovechar recursos distribuidos y establecer organizaciones de cooperación multi-institucional. Sin embargo, su aceptación general depende de que se adapte bien a diferentes esquemas de colaboración y de control de acceso a los recursos y datos que maneja cada institución.

En este trabajo presentamos algunos escenarios de acceso a datos entre instituciones que no han sido apropiadamente considerados hasta ahora, especialmente por el aspecto de confidencialidad de los datos. Se evaluaron los aspectos técnicos relevantes para la operación en estos escenarios para dos tipos de plataforma *grid*: SUMA/G y *grids* basados en Globus. Para los escenarios donde la confidencialidad de los datos impide que éstos sean agregados como recursos del *grid*, se propuso un esquema donde el acceso a los datos está bajo control del usuario autorizado de los datos, y no de los administradores del *grid*. Un esquema de implementación y uso de este esquema fue descrito en este trabajo.

En los escenarios donde se restringen las copias de los datos al *grid*, se mostró cómo los mecanismos de SUMA/G permiten adaptarse bien a ese contexto. Al combinarse con el esquema para acceso seguro a datos externos al *grid*, convierten a SUMA/G en una plataforma apropiada para su empleo en los escenarios de confidencialidad descritos en este trabajo. En contraste, se mostró cómo los *grids* actuales basados en Globus no pueden adaptarse fácilmente a escenarios que no admiten la copia de archivos de datos al *grid*.

Como trabajo futuro se plantea evaluar e implementar diferentes esquemas de integración de dominios de confianza en plataformas *grid*, tanto para plataformas basadas en Globus como en SUMA/G.

Referencias

[1] W. Allcock, J. Bresnahan, I. Foster, L. Liming, J. Link, and P. Plaszczac. GridFTP Protocol Specification. Technical report, Global Grid Forum, January 2002.

[2] Randy Butler, Von Welch, Douglas Engert, Ian Foster, Steven Tuecke, John Volmer, and Carl Kesselman. A national-scale authentication infrastructure. *Computer*, 33(12):60–66, 2000.

[3] Y. Cardinale and E. Hernández. Parallel Checkpointing on a Grid-enabled Java Platform. *Lecture Notes in Computer Science*, 3470(EGC2005):741 – 750, February 2005.

[4] Yudith Cardinale, Mariela Curiel, Carlos Figueira, Pedro García, and Emilio Hernández. Implementation of a CORBA-based metacomputing system. *Lecture Notes in Computer Science*, 2110, 2001. Workshop on Java in High Performance Computing.

[5] Globus Grid Forum. GT Data Management: Replica Location Service (RLS), 2006. http://www.globus.org/toolkit/data/rls/.

[6] Ian Foster and Carl Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*, chapter Computational Grids, pages 15–51. Morgan Kaufmann Publishers, Inc., 1999.

[7] Ian T. Foster, Carl Kesselman, Gene Tsudik, and Steven Tuecke. A security architecture for computational grids. In *ACM Conference on Computer and Communications Security*, pages 83–92, 1998.

[8] Neil M. Haller. The S/KEY one-time password system. In *Proceedings of the Symposium on Network and Distributed System Security*, pages 151–157, 1994.

[9] Yoshiyuki Kido, Susumu Date, Shingo Takeda, Shoji Hatano, Juncai Ma, Shinji Shimojo, and Hideo Matsuda. Architecture of a grid-enabled research platform with location-transparency for bioinformatics. *Genome Informatics*, 15(2):3–12, 2004.

[10] D. Kristol and L. Montulli. HTTP State Management Mechanism, February 1997. RFC 2109. http://www.ietf.org/rfc/rfc2109.txt.

[11] LCG Team. LCG: Worldwide LHC Computing Grid. http://lcg.web.cern.ch/lcg/, 2006.

[12] D. Mazieres. Security and decentralized control in the SFS global file system, Aug 1997. MIT Master's thesis.

[13] David Mazières. *Self-certifying file system*. PhD thesis, Massachusetts Institute of Technology, May 2000.

[14] L. Seitz, J. Montagnat, J.-M. Pierson, D. Oriol, and D. Lingrand. Authentication and autorisation prototype on the microgrid for medical data management. In *HealthGrid'05*, Oxford, UK, April 2005.

[15] The Globus Alliance. The Globus Toolkit, 2006. http://www.globus.org/.

[16] G. von Laszewski, I. Foster, J. Gawor, W. Smith, and S. Tuecke. CoG Kits: A Bridge between Commodity Distributed Computing and High-Performance Grids. In *ACM Java Grande 2000 Conference*, pages 97–106, San Francisco, CA, 3-5 June 2000.