Capítulo 6

Implementación de la infraestructura de autorización

El estudio realizado en el capítulo anterior acerca de los componentes de la infraestructura de autorización se centró principalmente en el proceso de diseño. Con el fin de comprobar la viabilidad de dichas propuestas, el capítulo que aquí nos ocupa se centrará en todos aquellos aspectos relacionados con la implementación e integración de la infraestructura en escenarios de aplicación concretos.

Dado que la mayor parte de los desarrollos que se plantearán están basados en la arquitectura CDSA (Common Data Security Architecture) [87], en primer lugar se realizará una introducción de las principales características de dicha arquitectura, introducción que estará enfocada principalmente a identificar los principales módulos de los que se ha hecho uso así como las limitaciones que se han encontrado a lo largo de la evolución de las implementaciones.

Posteriormente, se expondrá cómo se ha desarrollado tanto el marco AMBAR como el sistema DCMS. En ambos casos, el objetivo principal es presentar las librerías de programación y aplicaciones relacionadas, especialmente desde el punto de vista de su uso e integración con otros proyectos.

El siguiente paso consistirá en la descripción de dos escenarios de aplicación concretos en los cuales tanto la PKI basada en X.509 como el sistema DCMS se han utilizado para proporcionar mecanismos de autenticación y autorización. En concreto se trata de un entorno de control de acceso físico y un sistema de suscripción electrónica basada en un protocolo seguro de pagos.

Por último se realizará un análisis del rendimiento ofrecido tanto por la implementación de AMBAR como de DCMS. Dicho análisis tiene como objetivo extraer conclusiones acerca de la sobrecarga que pueden llegar a introducir dichos elementos dentro de cualquier escenario de autorización.

6.1 La arquitectura CDSA

En lo que respecta a arquitecturas de seguridad software, o también denominado *middleware* de seguridad, existen actualmente dos propuestas principales que intentan aglutinar los distintos algoritmos, protocolos y estándares de representación de información en lo que a criptografía y certificación digital se refiere.

Por un lado, el lenguaje de programación Java posee su propia arquitectura denominada JCA (Java Cryptography Architecture) [112], la cual está basada en la definición de un conjunto de interfaces y clases abstractas Java que tienen como objetivo definir una API (Application Programming Interface) de acceso a servicios de seguridad. La API JCA oculta a los programadores de aplicaciones de seguridad los detalles concretos de utilización de cada algoritmo criptográfico, elemento de información (certificados, solicitudes, listas de certificados revocados) o protocolo. Aunque la implementación concreta de cada técnica criptográfica o protocolo no forma parte de la propia arquitectura, sí se define cuál es la interfaz que deben cumplir los distintos proveedores de servicios criptográficos a la hora de poder insertar dicha implementación dentro del sistema.

Actualmente, existen multitud de proveedores que ofrecen conjuntos más o menos amplios de algoritmos y protocolos accesibles a través de la JCA, entre ellos IAIK [78], JCSI [55] o Cryptix [128]. Sin embargo, ninguno de estos proveedores proporciona soporte para las especificaciones de certificados de credencial analizadas en la sección 4.3, a excepción de IAIK que incluye un subconjunto muy reducido de las características de los certificados de atributo X.509. La especificación SPKI tiene asociadas varias implementaciones realizadas en este lenguaje de programación [147, 152, 161], todas ellas caracterizadas por ser totalmente independientes de la arquitectura JCA y carecer de soporte actualmente.

Junto con la falta de proveedores de certificados de credencial, otra de las carencias de JCA es la ausencia de características relacionadas con mecanismos de toma decisiones de autorización y de gestión de políticas. Al no ser parte de la arquitectura, dicha funcionalidad debe ser implementada de forma totalmente independiente, quizá por las propias aplicaciones finales, lo cual acaba propiciando que el sistema final derive en una composición no estructurada de elementos.

Por el contrario, la otra propuesta relevante en lo relativo a middleware de seguridad, la arquitectura CDSA, constituye un enfoque mucho más estructurado y ambicioso que viene a suplir la mayor parte de las carencias enunciadas anteriormente. Dado que todas las implementaciones han sido realizadas haciendo uso de CDSA, en los próximos apartados se introducirán los detalles más importantes de dicha arquitectura.

6.1.1 Visión general de la arquitectura CDSA

La arquitectura CDSA (Common Data Security Architecture) [87, 88] es un conjunto estructurado de servicios e interfaces de programación, expresadas en lenguaje C, encargado de proporcionar diversos mecanismos de seguridad a aplicaciones finales. Las capas inferiores de la arquitectura aportan los componentes fundamentales, tales como algoritmos criptográficos o generadores de números pseudo-aleatorios. Sobre dichos componentes se

construyen módulos relacionados con la implementación de certificados digitales, mecanismos de gestión de claves, certificados de credencial, gestión de políticas o almacenamiento de material criptográfico. Finalmente, en los niveles superiores se desarrollan los protocolos de seguridad de más alto nivel.

El diseño de la arquitectura sigue cinco principios estructurales:

- *Modelo de proveedores de servicio basado en niveles.* CDSA está constituida como un conjunto de niveles horizontales, los cuales proporcionan servicio a los niveles superiores.

- *Proceso abierto de diseño.* Es una propuesta abierta de Open Group [86], y como tal está expuesta a un proceso público de revisión y contribución.

- *Modularidad y extensibilidad.* Los componentes de cada nivel pueden utilizarse como si de módulos independientes se tratara. La arquitectura proporciona un marco mediante el cual introducir nueva funcionalidad o nuevas implementaciones más eficientes de servicios ya existentes.

- *Transparencia de cara al programador.* CDSA gestiona de forma interna la mayor parte de los detalles de seguridad relacionados con los servicios ofrecidos a las aplicaciones. De esta forma, los programadores de aplicaciones de alto nivel no necesitan conocer los pormenores de cada uno de los módulos que aportan funcionalidad al conjunto de la arquitectura.

- *Incorporación de tecnologías emergentes.* El diseño de la arquitectura está continuamente sujeto a revisiones motivadas por el surgimiento de nuevas tecnologías de seguridad a incorporar.

La figura 6.1 muestra los tres niveles básicos de CDSA: servicios de seguridad del sistema, gestor de servicios de seguridad comunes (CSSM, Common Security Services Manager) y módulos de seguridad incorporables (add-in).

El gestor CSSM es el núcleo de CDSA. CSSM gestiona cada una de las clases de servicios de seguridad ofrecidos así como las distintas implementaciones de dichos servicios mediante módulos *add-in*. En concreto, se encarga de la definición de la interfaz de programación (API, Application Programming Interface), la definición de la interfaz de proveedores de servicio (SPI, Service Provider Interface), y de la carga y ejecución dinámica de componentes. Respecto a esto último es importante recalcar que CSSM realiza un control de la integridad de todos los módulos de menor nivel cada vez que debe ejecutarse alguna de las funciones contenidas en dichos módulos. El proceso está basado en la verificación de la firma digital del código y su ejecución dentro de una base de computación confiable (TCB, Trusted Computing Base).

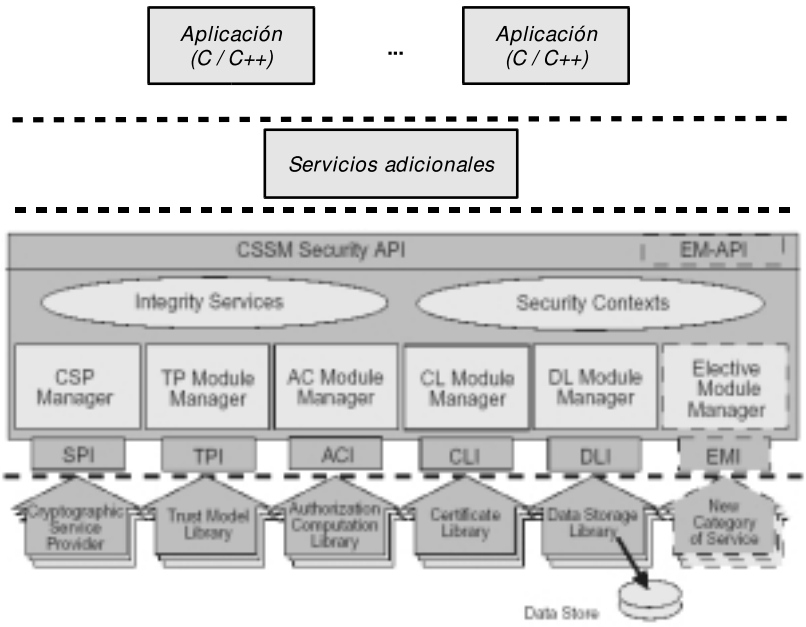

Figura 6.1: Arquitectura CDSA

6.1.2 Módulos de seguridad principales

En CDSA, todas las operaciones de seguridad deben ser realizadas finalmente por alguno de los módulos *add-in* instalados en el sistema. Dichos módulos se encuentran agrupados en cinco categorías de servicio:

- *Proveedores de servicios criptográficos.* Son módulos diseñados para realizar operaciones criptográficas y para almacenar de forma segura claves privadas. Un CSP puede implementar operaciones de criptografía simétrica y asimétrica, firma digital, resumen digital o generación e importación de claves.

- *Módulos de políticas de confianza.* Este tipo de módulos implementan las políticas definidas por las autoridades o instituciones. Dentro de CDSA, las políticas definen el nivel de confianza necesario para que ciertas acciones relacionadas con la certificación puedan llevarse a cabo.

- *Módulos de manejo de certificados.* Estos módulos implementan la manipulación sintáctica de los certificados y CRLs contenidos en memoria. Cada implementación de este tipo de módulos debe ofrecer las operaciones relacionadas con el tipo concreto de certificado al cual dé soporte, sea este X.509, SDSI, SPKI o cualquier otro. Las operaciones más comunes de manejo de certificados son la firma y verificación, la extracción de valores contenidos en campos concretos y la gestión de una CRL.

- *Módulos de almacenamiento de datos.* Este tipo de módulos ofrece operaciones para el almacenamiento persistente de información de seguridad. Dicha información pueden

ser certificados, listas de certificados revocados, claves criptográficas, credenciales, políticas u objetos dependientes de la aplicación.

- *Módulos de toma de decisiones de autorización.* Este módulo toma decisiones de autorización basadas en la política de seguridad del sistema evaluador, la solicitud cuya autorización se está evaluando y las credenciales presentadas por el solicitante.

Cada instancia o implementación de dichos módulos se instala en la arquitectura haciendo uso del servicio de directorio de módulos (MDS, Module Directory Service). Dicho servicio registra el identificador del módulo, una descripción de los servicios que éste proporciona y la información necesaria para cargar de forma dinámica el módulo. De esta manera, las aplicaciones pueden utilizar MDS para consultar qué módulo ofrece alguna de las funcionalidades necesarias para su ejecución.

6.1.3 Integración de CDSA en los desarrollos presentados

La implementación más completa de la arquitectura CDSA ha sido realizada por Intel [56]. Se trata de un proyecto de código abierto [182], periódicamente actualizado, que incorpora un gran número de módulos *add-in* para cada uno de los tipos de servicio ofrecidos por la arquitectura. Entre dichos módulos encontramos:

- *Proveedores de servicios criptográficos.* Hay disponibles dos proveedores distintos, uno de ellos basado en la librería criptográfica OpenSSL [167] y otro en BSAFE [180].

- *Módulos de manejo de certificados.* Tanto X.509 como SPKI disponen de su propio módulo CL (Certificate Library) para la construcción y uso de este tipo de certificados.

- *Módulos de gestión de confianza.* El único módulo incorporado implementa funciones de validación de certificados X.509.

- *Módulos de almacenamiento de datos.* La distribución proporciona una simulación de una base de datos implementada mediante ficheros (flat files).

- *Módulos de toma de decisiones de autorización.* La librería incluye un motor de cálculo de cadenas de certificación basado en la representación de tuplas comentada en la sección 4.3.4. Junto con el motor están disponibles funciones encargadas de realizar la conversión de certificados SPKI a tuplas.

Funcionalidad utilizada en los desarrollos

En los desarrollos que se describen en este capítulo se ha empleado un subconjunto de los módulos anteriormente enunciados. A continuación, se incluye una relación de dichos módulos y de los componentes de la infraestructura que han hecho uso de ellos:

- *Proveedor de servicios criptográficos basado en OpenSSL.* Las rutinas criptográficas han sido utilizadas de forma directa para la implementación del protocolo AMBAR (ver sección 6.2) y para los mecanismos de gestión de claves de las autoridades y claves temporales de usuario del sistema DCMS (ver sección 6.3).

- *Módulo de manejo de certificados X.509.* Este módulo se emplea tanto en la implementación de la fase de negociación SM del protocolo AMBAR como en algunas de las aplicaciones que forman parte del sistema DCMS.

- *Módulo de manejo de certificados SPKI.* Se utiliza en las implementaciones tanto de las aplicaciones de los solicitantes DCMS como de las autoridades de autorización y nombramiento.

- *Módulo de gestión de confianza.* Se emplea para la validación de certificados X.509 realizada durante la fase de negociación de AMBAR.

- *Módulo de toma de decisiones AuthCompute.* El motor de cálculo de autorizaciones se utiliza tanto en el proceso de optimización de solicitudes llevado a cabo por el módulo RM de AMBAR como en la validación de solicitudes de certificación realizada por las autoridades DCMS.

Como se verá en la sección 6.4, dentro del grupo de investigación ANTS se ha realizado también una implementación propia de un proveedor de servicios criptográficos y un módulo de almacenamiento de datos que permite integrar el uso de tarjetas inteligentes dentro de CDSA.

Limitaciones de CDSA

Durante el proceso de desarrollo de la infraestructura se identificaron varias carencias en la implementación de CDSA. Algunas de estas limitaciones eran propias del diseño de CDSA y otras formaban parte de los desarrollos llevados a cabo por Intel. A continuación se presenta una relación de las más importantes y se indica en qué estado se encuentra la corrección de las mismas.

- *Limitación en la generación de certificados.* La versión 3.12 de la implementación de CDSA desarrollada por Intel sólo proporcionaba mecanismos para generar certificados SPKI de autorización. En consecuencia, no era posible generar certificados de pertenencia a grupos (certificados de identidad) ni certificados de asignación de privilegios a nombres (certificados de atributo). Pocos meses después de la notificación que llevamos a cabo acerca del problema a los responsables del núcleo SPKI de CDSA se publicó la versión 3.14, la cual soportaba los tres tipos de certificados.

- *Error en el motor de toma de decisiones.* En Junio de 2002 identificamos un error bastante grave en el funcionamiento del módulo de cálculo de autorizaciones *AuthCompute*. El error permite a un usuario obtener privilegios para realizar cualquier

operación sobre un recurso determinado siempre que demuestre que ha sido autorizado a realizar al menos dos de ellas. A pesar de la notificación del error al equipo de CDSA, en el momento de la redacción de este trabajo, la deficiencia sigue sin subsanarse.

- *Limitación en la gestión de la confianza.* Pese a contar con un servicio de políticas de confianza, a la hora de integrar la gestión y el cumplimiento de los distintos tipos de políticas que forman parte de la infraestructura de autorización, se descubrió que la API correspondiente este tipo de módulos está demasiado enfocada a las operaciones de validación de certificados y listas de certificados revocados. En consecuencia, resulta prácticamente imposible integrar como parte de la arquitectura operaciones tan comunes como la verificación de solicitudes de certificación o la inserción y modificación de políticas. Sería aconsejable que se produjera un replanteamiento de la interfaz y los servicios a ofrecer por este tipo de módulos con el fin de poder adaptarlos a las nuevas tendencias relacionadas con políticas de seguridad.

A pesar de todo, CDSA puede considerarse el mejor *middleware* de seguridad existente para desarrollar servicios de autorización ya que proporciona un tratamiento integral a todos los aspectos que, de una forma u otra, se encuentran relacionados con este servicio. Además, su modelo basado en módulos extensibles favorece la incorporación de nuevas funcionalidades a la infraestructura con un número de modificaciones mucho menor del que se requeriría con otros enfoques menos estructurados.

6.2 Implementación del marco AMBAR

El marco AMBAR, y más concretamente el protocolo del mismo nombre, ha sido implementado como un servicio de seguridad basado en la arquitectura CDSA (versión 3.14 de Intel para Linux). Se trata de una librería de programación, desarrollada en lenguaje C++, que proporciona una interfaz mediante la cual es posible crear sesiones AMBAR tanto de cliente como de servidor.

El mecanismo de sesiones oculta los detalles relativos a la arquitectura interna del protocolo, sin por ello dejar de ofrecer toda la funcionalidad de cada uno de los módulos del marco. Del mismo modo, los programadores que hagan uso de la librería no necesitan tampoco estar familiarizados con los detalles de CDSA ya que todos los aspectos relativos a la criptografía y manejo de certificados forma parte de la implementación interna del protocolo.

Así pues, desde el punto de vista del programador de aplicaciones de alto nivel, sólo es necesario conocer dos aspectos de la implementación: por un lado, la API de acceso al protocolo, compuesta principalmente por los métodos relacionados con las sesiones AMBAR y el contexto de comunicación; por otro lado, la dinámica del protocolo, es decir, la secuencia de estados por los que puede atravesar una sesión. En esta sección, además de los dos detalles enunciados, se analizará también cómo se integra la implementación de AMBAR dentro de CDSA.

6.2.1 Integración de la arquitectura CDSA

La implementación de AMBAR está completamente basada en el uso de los servicios ofrecidos por la arquitectura CDSA, tal y como puede observarse en la figura 6.2.

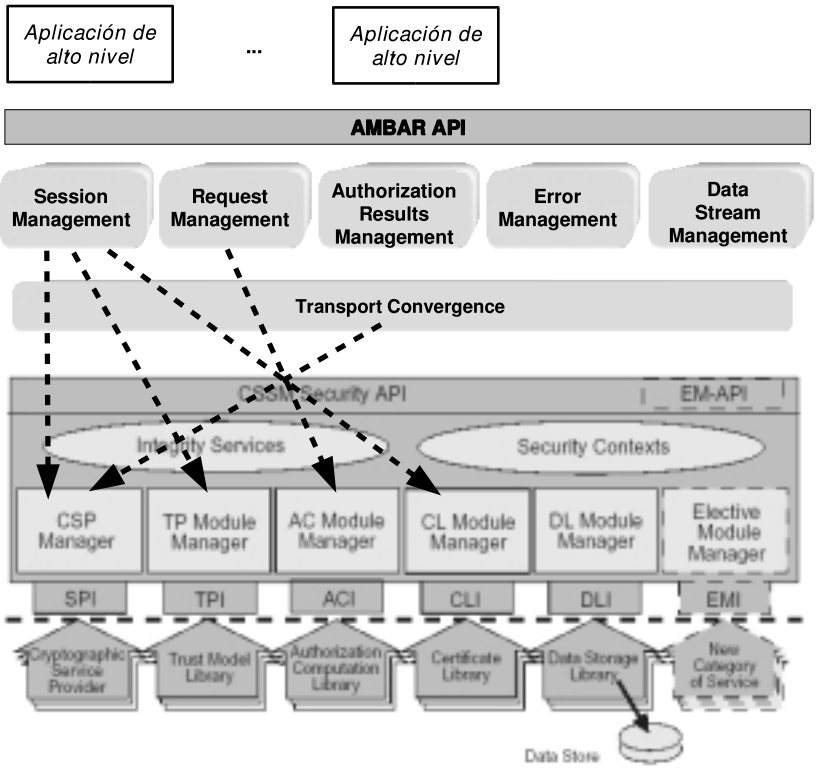

Figura 6.2: Integración de AMBAR con CDSA

Como se aprecia, AMBAR se encuentra ubicado en el nivel de servicios de seguridad construidos sobre la arquitectura. Las aplicaciones de alto nivel se comunican con la librería mediante una API que simplifica el uso del marco y oculta todos aquellos detalles irrelevantes para los programadores de aplicaciones de control de acceso a recursos.

Se observa también que los módulos AMBAR emplean la mayor parte de los proveedores de servicio contenidos en la implementación elegida de CDSA. Más concretamente, los servicios utilizados por cada módulo son:

- *Módulo SM.* Usa el módulo CSP (Cryptographic Service Provider) para realizar las operaciones de criptografía asimétrica relacionadas con los mensajes de autenticación (concretamente el mensaje *ActivateCrypto*), así como para generar las cargas aleatorias intercambiadas al principio de la fase. Además utiliza los módulos CL (Certificate Library) y TP (Trust Policy) para procesar y validar los certificados X.509 intercambiados.

- *Módulo RM.* Emplea el módulo AC (AuthCompute) para construir las reducciones de los certificados intercambiados.

- *Nivel TC*. Mediante el módulo CSP, realiza las transformaciones criptográficas de los mensajes AMBAR generados por los módulos superiores. En concreto, utiliza rutinas de cifrado simétrico y de construcción de códigos de autenticación.

6.2.2 Esquema de funcionamiento del protocolo

A la hora de emplear el protocolo como mecanismo de control de acceso a recursos protegidos, es necesario conocer cuál es la secuencia de estados por la que puede atravesar, es decir, cuáles son todas las posibilidades que pueden llegar a darse en una comunicación entre un controlador y un solicitante. Este conocimiento del funcionamiento del protocolo permite al programador comprender el uso correcto de la interfaz de programación ofrecida, es decir, de las primitivas de servicio ofrecidas por la librería para que esta pueda emplearse como si se tratase de un nivel de abstracción dentro de una arquitectura de red. La secuencia de estados permite conocer la dinámica del protocolo, y por tanto el orden en el cual deben ejecutarse los distintos métodos que forman parte de la API.

En la figura 6.3 se muestra un autómata de estados en el que cada transición representa la recepción o el envío de un mensaje AMBAR por parte de un solicitante. Aunque la representación está enfocada desde el punto de vista del cliente, es lo suficientemente completa como para orientar la programación de aplicaciones AMBAR de servidor.

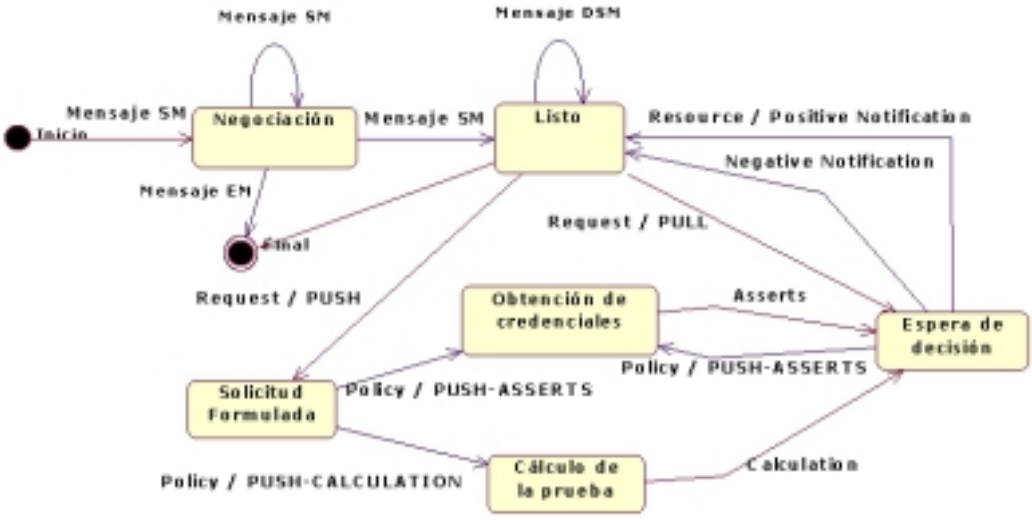

Figura 6.3: Secuencia de estados de AMBAR

En primer lugar se produce el inicio de la fase de negociación, la cual representa el primer estado del autómata. Por simplicidad se han agrupado todos los mensajes de esta fase con el nombre genérico de *mensaje SM*. Dado que la negociación se realiza de forma transparente para las aplicaciones de nivel superior, no es necesario detallar los estados que forman parte de esta fase. En el caso de que se produzca un error en la verificación de un mensaje o se presenten parámetros de autorización incompatibles se llega al final de la ejecución (estado *Final*).

Si la negociación finaliza con éxito, la ejecución se sitúa en el estado *Listo*. A partir de este instante, la posible secuencia de estados varía completamente en función del método de distribución de información negociado. A continuación se desglosan las tres opciones válidas:

- *Distribución pull*. Con esté método, tras el envío del mensaje *Request* el solicitante queda a la espera de que el controlador envíe alguno de los tres tipos de mensajes ARM (recurso y notificación negativa o positiva). En los tres modos de distribución, la recepción de un mensaje ARM implica una vuelta al estado *Listo*.

- *Distribución push-asserts*. Este método de distribución es propio de los controladores que realizan una revelación progresiva de su política de seguridad. Tras el envío de la solicitud se llega al estado *Solicitud formulada* y se produce la recepción del primer mensaje *Policy*, el cual implicará la transmisión de un conjunto de credenciales relacionadas con la política recibida. El número de veces que se repite dicho intercambio de mensajes *Policy* y *Asserts* depende de la estrategia del controlador en cuestión, y finaliza con la transmisión por parte del controlador de alguno de los mensajes ARM.

- *Distribución push-calculation*. Este caso se diferencia del anterior en el hecho de que la recepción del mensaje *Policy* no conlleva el envío de credenciales, sino que lleva al autómata al estado *Cálculo de la prueba*. Dicho estado implica el descubrimiento de una cadena de certificación que demuestre las posibilidades de acceso del solicitante según lo formulado por la política recibida. Tras el envío de dicha cadena al controlador, el solicitante debe quedar a la espera de un mensaje ARM.

Aunque no se muestran en la figura, la recepción o el envío de la mayor parte de mensajes EM implica el fin prematuro de la ejecución del protocolo. De hecho, desde cualquiera de los estados mostrados en la figura 6.3 habría una transición hacia un estado de finalización anormal.

Por último, comentar que la implementación del protocolo aquí presentada controla que la ejecución se ajuste rigurosamente al autómata presentado. Cualquier intento de transición no contemplada, motivada por el envío o recepción de mensajes incorrectos, es detectado e implica la finalización inmediata de la comunicación.

6.2.3 Interfaz de programación (API)

La API de acceso a la librería AMBAR está formada principalmente por 4 clases distintas: *AMBARClientSession*, *AMBARServerDaemon*, *AMBARServerSession* y *AMBARContext*. La relación entre dichas clases puede verse en el diagrama de clases UML [24] que muestra la figura 6.4.

AMBARContext

La construcción del *AMBARContext* es la primera acción que debe llevarse a cabo antes de establecer la comunicación. Esta clase representa el contexto de comunicación AMBAR, es

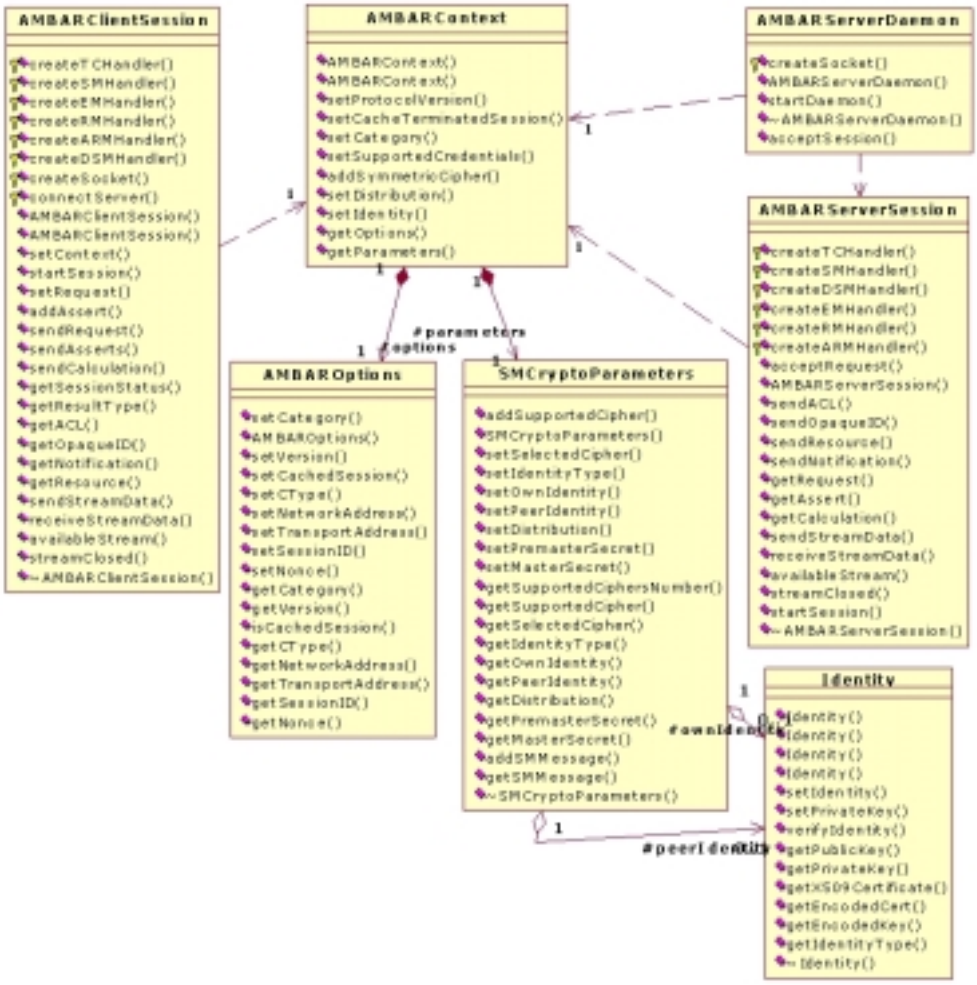

Figura 6.4: Clases de la API de AMBAR

decir, todos aquellos elementos de información relacionados con los parámetros de autorización a negociar, identidad de las entidades participantes y material criptográfico generado durante la sesión. Su implementación está basada en la agregación de dos objetos pertenecientes a las clases *AMBAROptions* y *SMCryptoParameters*, los cuales proporcionan la mayor parte de la funcionalidad de *AMBARContext*.

Si bien la cantidad de elementos de información gestionada por esta clase es muy elevada, el programador de aplicaciones tanto de cliente como de servidor sólo debe conocer los métodos principales relacionados con la especificación de los parámetros de autorización a negociar:

- *setProtocolVersion()*. Establece la versión del protocolo AMBAR que se desea utilizar.

- *setCategory()*. Indica el modo de operación preferido (anónimo o identificado).

- *setSupportedCredentials()*. Se utiliza para especificar el tipo de certificados de credencial que será empleado para determinar los privilegios.
- *addSymmetricCipher()*. Mediante este método se introducen, por orden de prioridad, los algoritmos simétricos preferidos.
- *setDistribution()*. Establece el método de distribución de credenciales a emplear en la sesión (*pull*, *push-asserts* y *push-calculation*).
- *setIdentity()*. Se utiliza tanto para especificar el certificado de identidad del comunicante como para establecer el conjunto de autoridades de certificación confiables.

Una vez creado el contexto, éste debe pasarse a las clases que representan las sesiones para que puedan emplearlo durante la fase de negociación.

AMBARClientSession

La recepción y envío de mensajes por parte de los solicitantes se hace a través de *AMBARClientSession*. Esta clase encapsula todos los aspectos relacionados con el establecimiento de la comunicación TCP/IP, la negociación de sesiones, el envío y optimización de solicitudes, y la recepción de notificaciones procedentes del controlador. Sus métodos se pueden agrupar en los siguientes bloques funcionales:

- Métodos relacionados con el establecimiento de los parámetros de comunicación y de autorización:
  - *AMBARClientSession()*. Mediante el constructor es posible especifica la dirección IP y el puerto TCP del controlador elegido.
  - *setContext()*. Establece el contexto AMBAR de la sesión.
- Métodos relacionados con la fase de negociación de sesiones:
  - *startSession()*. La llamada a este método implica el inicio de la fase de negociación AMBAR.
- Métodos relacionados con el envío de solicitudes, políticas y credenciales:
  - *setRequest()*. Establece una nueva solicitud de acceso.
  - *addAssert()*. Mediante este método es posible incluir credenciales en el próximo mensaje a enviar por el solicitante.
  - *sendRequest()*. Envía la solicitud de acceso y, si se produjo alguna llamada a *addAssert*, las posibles credenciales asociadas.
  - *sendAsserts()*. Envía un conjunto de certificados de credencial.

  - *sendCalculation()*. Envía una prueba de autorización calculada previamente por el solicitante.
  - *getSessionStatus()*. Este método indica si el controlador ha tomado una decisión definitiva acerca de la solicitud en curso o si por el contrario ha enviado información relacionada con su política de control de acceso, es decir, si le está pidiendo al solicitante más evidencias.
  - *getACL()*. Con este método es posible obtener la última política enviada por el controlador durante la transacción en curso.

- Métodos relacionados con la obtención de respuestas del controlador:

  - *getResultType()*. Este método indica si la respuesta es una notificación (positiva o negativa) o bien el recurso solicitado.
  - *getNotification()*. En el caso de que el controlador haya enviado una notificación, este método devuelve el contenido de la misma.
  - *getResource()*. Devuelve el contenido del recurso solicitado.

- Métodos relacionados con la gestión de flujos de datos:

  - *sendStreamData()*. Una vez que ha sido autorizada una solicitud relacionada con el establecimiento de un flujo de datos entre controlador y solicitante, este método se emplea para enviar información que será transportada sin modificación alguna hasta el controlador. Dicha información puede tratarse de datos de un protocolo de nivel superior.
  - *receiveStreamData()*. Este método devuelve parte de los datos enviados por el controlador a través del flujo.
  - *availableStream()*. Indica el número de bytes pendientes de lectura que están almacenados en el buffer de recepción del flujo.
  - *streamClosed()*. Mediante este método es posible conocer si el flujo sigue activo o si ha sido cancelado por la otra entidad.

Como puede apreciarse estudiando el API ofrecida, el programador de aplicaciones cliente sólo debe tener conocimiento de la secuencia de estados por la que puede atravesar el protocolo (ver figura 6.3). El resto de detalles relacionados con el funcionamiento interno del protocolo, como el intercambio concreto de mensajes o la optimización de las solicitudes, no son visibles al programador. El resultado es una API bastante intuitiva que permite desarrollar de forma sencilla aplicaciones que hagan uso de AMBAR.

AMBARServerDaemon

Con el fin de que un mismo controlador pueda atender de forma simultánea peticiones de distintos solicitantes, la librería proporciona una clase con funcionalidad similar a la de un *daemon* de comunicaciones. La clase *AMBARServerDaemon* permite que los controladores puedan establecer varias sesiones simultáneas caracterizadas por los mismos parámetros de autorización, es decir, por el mismo contexto AMBAR. Una vez creado dicho contexto, es posible iniciar un *AMBARServerDaemon* y establecer sesiones haciendo uso de los siguientes métodos:

- *AMBARServerDaemon()*. Mediante el constructor es posible especificar tanto el puerto TCP asociado al controlador como el contexto AMBAR.
- *startDaemon()*. Inicia la ejecución del *daemon*.
- *acceptSession()*. Tras ejecutar este método el controlador queda a la espera del establecimiento de una nueva sesión.

Una conexión entrante realizada por parte de un solicitante implica la generación de una nueva sesión AMBAR de servidor mediante la cual será posible llevar a cabo el intercambio de la información relativa a autorización.

AMBARServerSession

Los objetos de esta clase no debe generarlos el programador ya que es el método *acceptSession()* de *AMBARServerDaemon* el encargado de generar las nuevas sesiones de servidor. La consecuencia es que el contexto de cada sesión ya está creado tomando como base el contexto del *daemon*. Por tanto, todos los métodos de esta clase están relacionados sólo con el intercambio de información entre solicitante y controlador:

- Métodos relacionados con la fase de negociación de sesiones:
  - *startSession()*. La llamada a este método implica el inicio de la fase de negociación AMBAR.
- Métodos relacionados con el intercambio de solicitudes, políticas y credenciales:
  - *acceptRequest()*. La ejecución de este método implica la espera de una petición de acceso procedente del solicitante.
  - *getRequest()*. Se utiliza para obtener la solicitud recibida.
  - *getAssert()*. Mediante este método es posible ir obteniendo las credenciales que ha enviado el solicitante, tanto en la solicitud como de forma independiente.
  - *sendACL()*. Envía la política relacionada con la solicitud recibida. El contenido de dicha política depende de la estrategia de revelación seguida por el controlador.

  - *getCalculation()*. Recibe la prueba de autorización hallada por el solicitante.

- Métodos relacionados con el envío de respuestas a las solicitudes:
  - *sendNotification()*. Devuelve una notificación, negativa o positiva, acerca del resultado del procesamiento de la petición formulada por el solicitante.
  - *sendResource()*. Envía el recurso protegido que se había solicitado.

- Los métodos relacionados con la gestión de flujos de datos son los mismos que ofrece la clase *AMBARClientSession*.

Como se verá en la siguiente sección, el protocolo AMBAR se ha integrado como parte de las aplicaciones de las autoridades y de los solicitantes del sistema DCMS. La librería ofrece toda la funcionalidad necesaria para implementar el intercambio de información entre las dos aplicaciones.

6.3 Implementación de DCMS

El primer paso en el desarrollo de DCMS fue la elección del tipo de implementación que se deseaba realizar, ya que había dos alternativas claras en lo que a la concepción final del sistema se refería. Por un lado, cabía la posibilidad de desarrollar aplicaciones que intentaran ocultar todos los detalles internos del sistema a los usuarios de las mismas, lo cual representaría la creación de herramientas de alto nivel totalmente independientes de cuestiones como el formato de las políticas, solicitudes, tags, etc. Por otro lado, existía la alternativa de crear herramientas más cercanas a la estructura del sistema, que permitieran ver y configurar todos los parámetros y elementos de información que forman parte de DCMS. Tomando como referencia la idoneidad para poder realizar un análisis y evaluación de las posibilidades de DCMS, finalmente se tomó la decisión de desarrollar aplicaciones altamente configurables que mostraran el mayor número posible de las características propias del sistema, como la monitorización de las comunicaciones realizadas, la visualización del contenido de todos los elementos de información o la configuración de los parámetros criptográficos.

Por otro lado, el conjunto de aplicaciones debía estructurarse de acuerdo con la metodología analizada en la sección 5.4. El objetivo era que la aplicación de la metodología tuviera una correspondencia directa con los distintos componentes del sistema, lo cual permitiría que la aplicación de los procedimientos contenidos en la metodología pudiera realizarse de forma estructurada y bien definida. Así pues, el resultado final fue que parte de las aplicaciones desarrolladas puede interpretarse como una materialización práctica de los distintos niveles de procedimientos. Dichas aplicaciones están claramente relacionadas entre sí y con las dos herramientas principales del sistema: la aplicación de las autoridades y la de los solicitantes. A lo largo de esta sección, se verá cuál es la funcionalidad de cada uno de los componentes del sistema así como algunos detalles concretos de la implementación del mismo.

6.3.1 Visión general

Todas las aplicaciones desarrolladas poseen su propia interfaz gráfica de usuario (GUI, Graphic User Interface), lo cual permite interactuar con ellas de forma amigable y visualizar la información ordenadamente. En concreto, su implementación se ha llevado a cabo en Linux, utilizando el lenguaje de programación C++, la arquitectura CDSA y las librerías gráficas QT [185]. La figura 6.5 muestra tanto la relación existente entre todas las aplicaciones del sistema DCMS que se han desarrollado como su interconexión con la infraestructura de clave pública y el marco AMBAR. Por un lado, dicha interconexión está basada en el uso de los servicios de generación y validación de claves proporcionados por la PKI, los cuales representan el proceso de gestión básico de las claves a las cuales se les asignarán las autorizaciones. Por otra parte, la implementación del marco AMBAR permite realizar el intercambio de información entre las autoridades y los solicitantes.

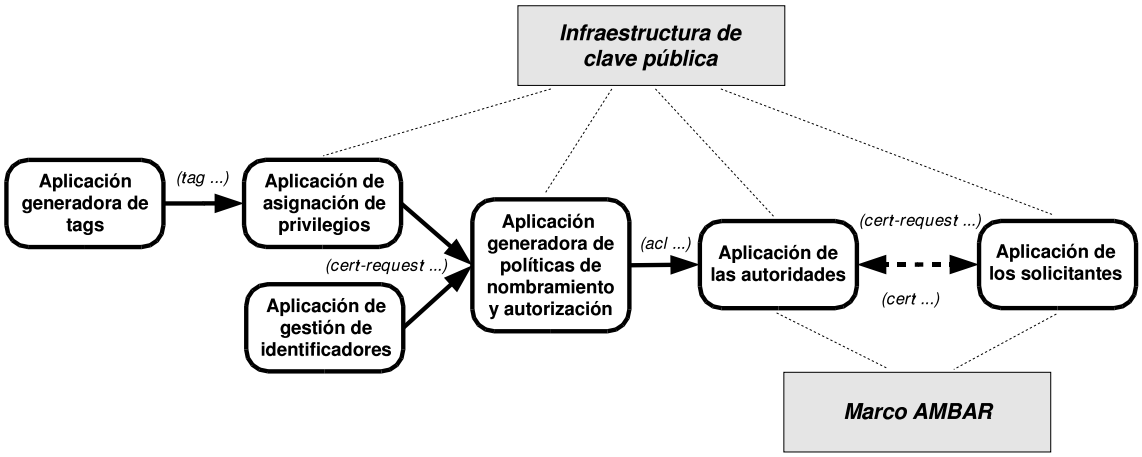

Figura 6.5: Conjunto de aplicaciones DCMS

Como se observa, los resultados producidos por parte de las aplicaciones del sistema DCMS suponen el punto de partida de otras, lo cual desemboca en una composición secuencial de elementos claramente relacionados entre sí. La figura muestra dos bloques de aplicaciones distintos ya que, por un lado, tenemos un conjunto de herramientas encargadas de producir las políticas de autorización y de nombramiento de las autoridades, y por otro lado se aprecian las dos aplicaciones que interactúan para generar los certificados de credencial SPKI (autoridades y solicitantes). En los siguientes apartados se realizará una descripción concreta de la funcionalidad de cada uno de los componentes de la figura.

6.3.2 Aplicación generadora de tags

La primera aplicación está relacionada con parte de los procedimientos contenidos en el nivel 0 de la metodología, concretamente con aquellos asociados a la especificación de los recursos que se desea proteger. Como se vio en la sección 5.4.1, dicha especificación

comprende la definición de un esquema de identificación de recursos y la identificación de las operaciones realizadas sobre dichos recursos.

La aplicación generadora de tags es totalmente dependiente del entorno de aplicación al cual se esté aplicando el sistema. Su objetivo principal es la creación de s-expresiones que representen tags de autorización válidos, los cuales serán posteriormente utilizados tanto para la definición de políticas de autorización como para la construcción de solicitudes de certificación. La sección 6.4 mostrará un ejemplo concreto de este tipo de aplicación.

Dejar constancia de que se trata del único componente del sistema DCMS que varía de un entorno de aplicación a otro. El resto de componentes, al estar basados en la composición de las s-expresiones que cada uno de ellos produce, no necesita ningún tipo de modificación para ser adaptado a otro escenario. De esta forma se minimiza el trabajo de desarrollo necesario para integrar DCMS en distintos sistemas.

6.3.3 Aplicación de asignación de privilegios

La aplicación de asignación de privilegios permite plasmar los procedimientos de nivel 2 del bloque AMS de la metodología. Se trata del componente mediante el cual es posible especificar quienes son las entidades autorizadas a obtener los privilegios previamente especificados mediante la aplicación comentada en el apartado anterior. La figura 6.6 muestra parte de la interfaz gráfica de este componente.

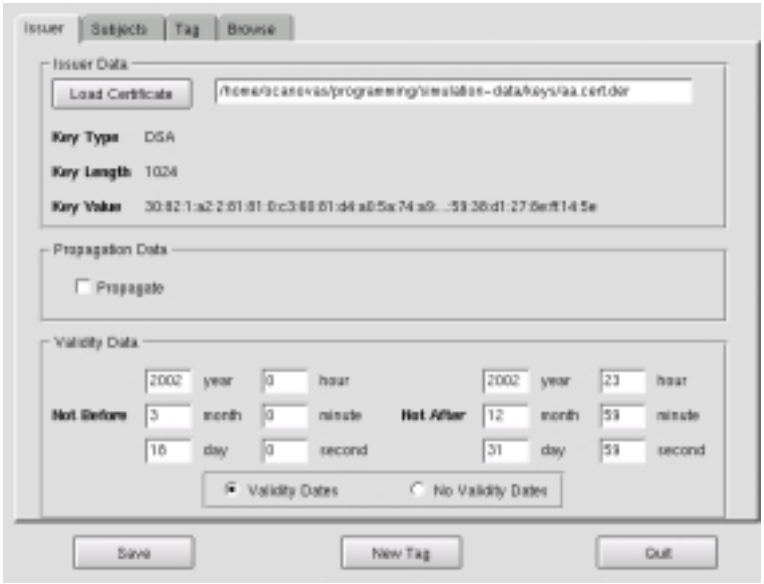

Figura 6.6: Aplicación de asignación de privilegios

La aplicación permite construir las s-expresiones del tipo *cert-request* que posteriormente formarán parte de la política de autorización de una autoridad concreta. Para ello, es necesario especificar:

- *Emisor de los certificados.* Se trata de la clave pública de la autoridad encargada de emitir los privilegios.
- *Receptores de los certificados.* Son todas aquellas claves públicas y nombres SDSI que están autorizados a recibir un conjunto determinado de privilegios.
- *Posibilidad de propagación de los privilegios.* Es posible especificar si el conjunto de entidades anterior tiene concedido el derecho a poder propagar a su vez los privilegios recibidos, es decir, a actuar como autoridades de autorización.
- *Privilegios asignados.* Se trata del resultado de la aplicación del apartado anterior, es decir, de s-expresiones que contienen tags de autorización concretos relacionados con el entorno de aplicación que se está modelando.
- *Periodo de disfrute de los privilegios.* Por último, se especifica el periodo de tiempo durante el cual los receptores podrían hacer uso de los privilegios concedidos.

Como resultado final se obtiene una s-expresión que codifica los criterios de asignación de privilegios seguidos por la entidad emisora. La s-expresión, cuya estructura fue analizada en la sección 5.3.4, es uno de los elementos de información necesarios para la aplicación generadora de políticas.

6.3.4 Aplicación de gestión de identificadores

La aplicación de gestión de identificadores está ligada al nivel 2 del bloque NMS de la metodología. Una vez que el primer nivel determina los elementos que se encuentran dentro del ámbito de la autoridad, es decir, el subconjunto de entidades individuales y los roles ya definidos del sistema, el nivel 2 debe determinar los identificadores que pueden ser asignados a cada uno de ellos. Esta asignación de identificadores se lleva a cabo mediante la aplicación mostrada en la figura 6.7.

La aplicación permite construir las s-expresiones del tipo *cert-request* que posteriormente formarán parte de la política de nombramiento de una autoridad concreta. Para ello, es necesario especificar:

- *Emisor de los certificados.* Se trata de la clave pública de la autoridad encargada de emitir los certificados de identidad SPKI.
- *Receptores de los certificados.* Son todas aquellas claves públicas y nombres SDSI que están autorizados a recibir un conjunto determinado de privilegios. En el caso de que se desee asignar nombres localmente únicos a entidades, el conjunto de receptores estará formado por una única clave pública. Si lo que se desea es gestionar la pertenencia a un rol, el conjunto de receptores puede estar formado tanto por las claves públicas de las entidades pertenecientes al mismo como por el nombre SDSI de los roles que se encuentran contenidos dentro del que se está definiendo (lo que conlleva la generación de jerarquías).

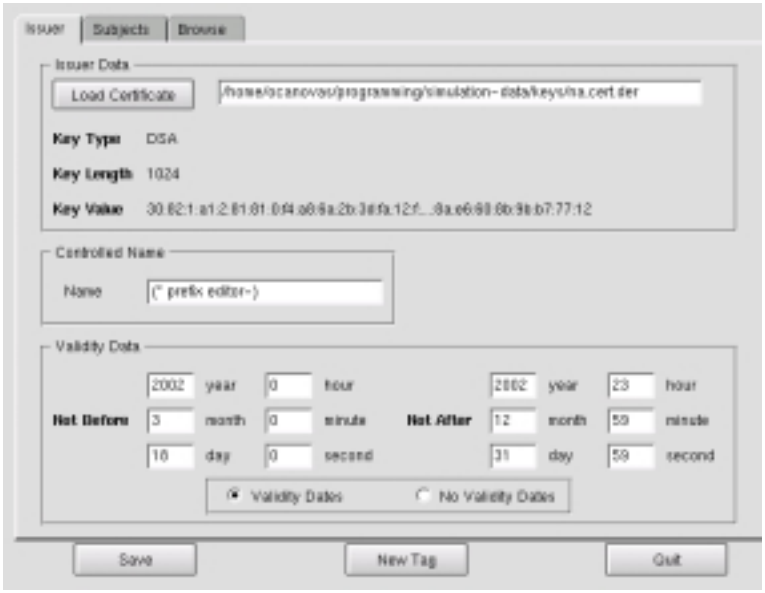

Figura 6.7: Aplicación de gestión de identificadores

- *Identificador a gestionar.* Se trata del identificador que será asociado a los receptores, el cual puede tratarse de un nombre de rol o de un identificador de usuario localmente único.

- *Periodo de asociación.* Por último, se especifica el periodo de tiempo durante el cual los receptores podrían permanecer ligados al identificador.

6.3.5 Aplicación generadora de políticas

La aplicación generadora de políticas es la aplicación práctica de los procedimientos de nivel 3 tanto del bloque AMS como NMS. Es decir, una vez realizada la asignación de privilegios e identificadores, mediante esta aplicación es posible finalizar la especificación de la política mediante el establecimiento del conjunto de solicitantes válidos y del periodo de solicitud. La figura 6.8 muestra parte de la interfaz gráfica de este componente.

Partiendo de las s-expresiones generadas mediante la aplicación de asignación de privilegios y la aplicación de gestión de identificadores, es posible especificar los siguientes parámetros:

- *Conjunto de solicitantes autorizados.* Se trata de las claves públicas o nombres SDSI de las entidades que están autorizadas a solicitar las credenciales. En el caso de que se omita se asumirá que coincide con el conjunto de entidades receptoras.

- *Posibilidad de delegación.* Es posible especificar si el conjunto de entidades solicitantes puede delegar el privilegio de solicitar las credenciales.

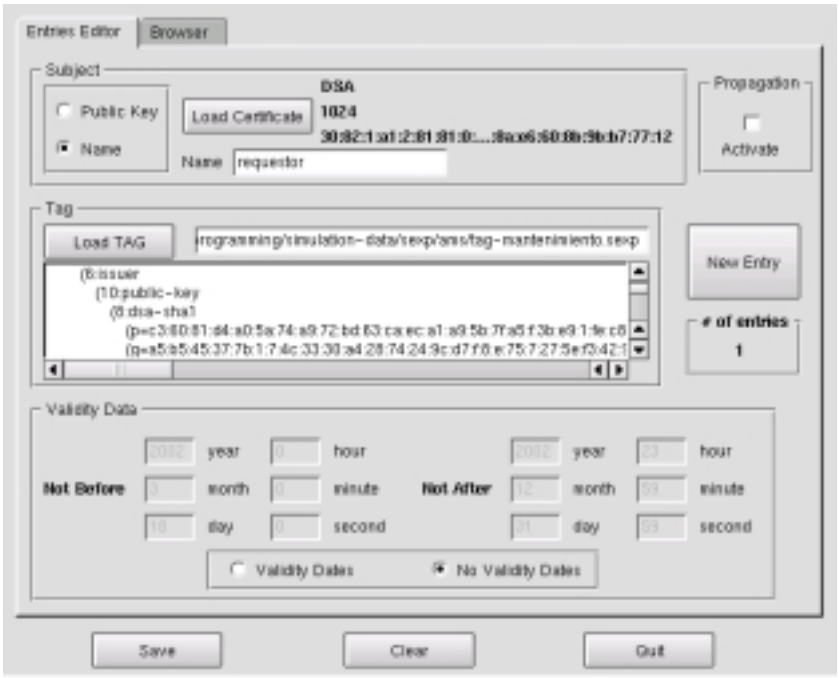

Figura 6.8: Aplicación generadora de políticas

- *Credenciales a emitir.* S-expresiones que denotan qué privilegios o qué identificadores pueden ligarse a un conjunto de entidades finales. En función del tipo de s-expresión, obtenidas como resultado de las aplicaciones de asignación de privilegios o de gestión de identificadores, se estará definiendo una política de autorización o una de nombramiento.

- *Periodo de solicitud.* Especifica el intervalo de tiempo durante el cual se puede llevar a cabo la solicitud de las credenciales. En caso de omisión se asumirá que coincide con el periodo de validez especificado en las s-expresiones anteriores.

El resultado principal de esta aplicación es la construcción de elementos de información muy similares a las listas de control de acceso SPKI que codifican todos los criterios de autorización derivados a partir de la aplicación de la metodología. Como se verá en el siguiente apartado, dichos elementos de información condicionan el comportamiento que tendrán las autoridades frente a las solicitudes de certificación recibidas de forma directa o bien a través de los puntos de acceso.

6.3.6 Aplicación de las autoridades

La aplicación de las autoridades constituye la herramienta más completa del sistema DCMS ya que puede actuar como autoridad de autorización, autoridad de nombramiento y reductor confiable. Además está diseñada para operar tanto en línea como desconectada de la

red. Sus usuarios serán todas aquellas entidades que tras la aplicación de los procedimientos de nivel 0 de la metodología deban asumir las funciones de gestión de un conjunto de credenciales. En los siguientes apartados se analizará de forma independiente cada uno de los grupos de operaciones que es posible realizar haciendo uso de ella.

Configuración de las propiedades de la autoridad

El primer bloque está relacionado con el establecimiento de las propiedades de la autoridad, entendiendo como propiedades los siguientes elementos:

- *Tipo de autoridad.* Para que la aplicación ofrezca sólo las funcionalidades asociadas a cada tipo de autoridad, en primer lugar es necesario especificar si actuará como autoridad de autorización, autoridad de nombramiento o reductor confiable.

- *Valores criptográficos.* Para poder emitir certificados es necesario especificar el par de claves asimétricas pertenecientes a la autoridad. Dicha configuración puede realizarse de tres formas distintas: en primer lugar es posible solicitar que la aplicación genere un nuevo par de claves criptográficas y un certificado X.509 autofirmado para la autoridad; por otro lado, también es posible generar el par de claves y construir una solicitud PKCS#10 con el fin de que el certificado de la autoridad sea generado por parte de la PKI; por último, se puede especificar la localización de una clave privada y certificado ya existentes.

- *Certificados digitales de la interfaz AMBAR.* Tal y como se comentó en la sección 5.3.6, no resulta conveniente que las autoridades utilicen sus claves privadas de firma para establecer conexiones AMBAR. Constituye una alternativa más correcta que éstas generen pares de claves temporales destinadas a proteger las comunicaciones. En consecuencia, la aplicación permite generar el par de claves temporales que será empleado para establecer sesiones AMBAR con los solicitantes y puntos de acceso. Para que dichas claves sean consideradas como válidas por el resto de entidades del sistema, la aplicación emite también un certificado de autorización que les confiere el privilegio de actuar como su interfaz AMBAR.

Configuración de los parámetros AMBAR

En relación con el modo de operación en línea de la autoridad, es necesario especificar las preferencias relacionadas con el establecimiento de sesiones AMBAR. En concreto, es necesario especificar:

- *Parámetros a negociar.* La aplicación permite especificar las preferencias de la autoridad en lo que respecta al tipo de certificados de identidad a emplear, tipo de certificados de credencial, método de distribución de credenciales, modo de comunicación y algoritmo de cifrado simétrico preferido. Estas preferencias están relacionadas con la aplicación de los procedimientos del nivel 4 de los bloques AMS y NMS de la metodología.

- *Parámetros de red.* Es posible especificar tanto el puerto TCP mediante el cual se realizará la comunicación con las entidades externas como el número máximo de sesiones simultáneas activas que se atenderán.

Especificación de las políticas

Uno de los modos de operación de las autoridades es la tramitación de solicitudes de certificación que conforman con la política de la autoridad. Por tanto, una de las opciones que ofrece la aplicación de las autoridades es la especificación de un conjunto indefinido de políticas que deben cumplirse, conjunto que podrá ser modificado en cualquier instante.

Operaciones en modo desconectado

La aplicación permite realizar varias operaciones de gestión en modo *desconectado*, es decir, no propiciadas por la recepción a través de un canal AMBAR de una solicitud de servicio. Esta opción es especialmente útil para gestionar privilegios que necesitan una protección especial. Más concretamente, las operaciones de este tipo son:

- *Generación de certificados.* Esta operación pueden realizarla tanto las autoridades de nombramiento como las de autorización. Se trata de un formulario que permite introducir todos los campos que forman parte de alguno de los tres tipos de certificados SPKI. De esta forma es posible emitir manualmente certificados que no estén contemplados en la política.

- *Reducción de certificados.* La operación de reducción está disponible tanto para los reductores confiables como para las autoridades de autorización. Como muestra la figura 6.9, la aplicación permite simplificar cadenas de certificación que tengan como raíz la clave pública de la autoridad o del reductor y como nodo final la clave pública de la entidad especificada en el formulario. Además, la reducción se puede restringir a un conjunto determinado de privilegios.

- *Gestión de solicitudes.* Esta operación, disponible para los tres tipos de autoridades, permite tramitar una solicitud existente. Dicha solicitud puede ser tanto de certificación como de reducción de una cadena de certificación. Para su tramitación se comprueba si cumple con la política de la autoridad, en cuyo caso se procede a la generación del certificado correspondiente.

Operaciones en línea

La aplicación permite activar un proceso independiente encargado de atender de forma concurrente, mediante una ejecución multihilo, las peticiones formuladas por los usuarios a través de la red, bien de forma directa o a través de los puntos de acceso. En todo momento, tal y como se muestra en la figura 6.10, es posible monitorizar cuál es la evolución de dichas comunicaciones. En concreto, la aplicación permite:

Figura 6.9: Reducción de certificados en modo desconectado

- Visualizar en qué instante se produjo el inicio de una negociación AMBAR o la tramitación de un solicitud para cada una de las conexiones entrantes activas.

- Conocer los parámetros de autorización negociados en cada una de las sesiones establecidas.

- Conocer el número total de solicitudes tramitadas correctamente y los datos relativos a los certificados generados en consecuencia.

En este modo, la emisión de certificados se realiza siempre tras la validación de que las solicitudes correspondientes cumplen la política especificada por la autoridad.

6.3.7 Aplicación de los solicitantes

Este componente recibe el nombre genérico de *aplicación de los solicitantes* porque aglutina tanto la funcionalidad de un punto de acceso como la de una aplicación de gestión de autorizaciones diseñada para usuarios finales. Su objetivo general es la provisión de mecanismos que permitan a las entidades finales crear y transmitir solicitudes de certificación y de reducción. En los siguientes apartados se analizará de forma independiente cada uno de los grupos de operaciones que es posible realizar haciendo uso de ella.

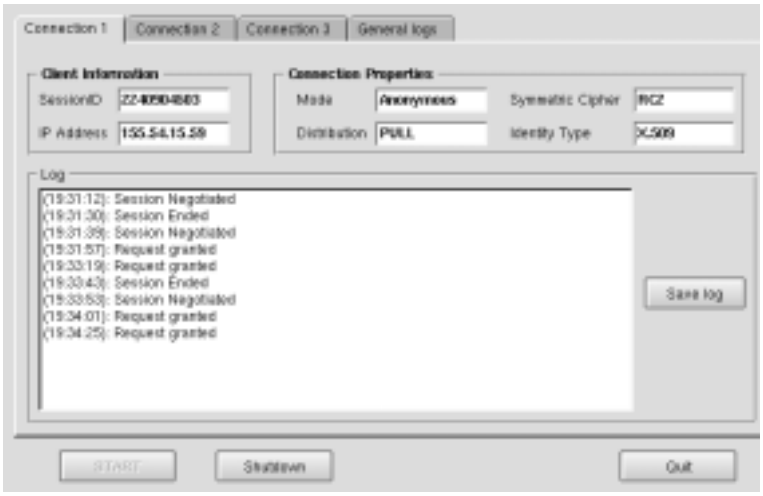

Figura 6.10: Monitorización de las conexiones de la autoridad

Gestión de claves temporales

Tal y como se comentó en la sección 4.4.3, el uso de claves temporales por parte de los usuarios finales proporciona dos servicios adicionales de especial utilidad. Por un lado, un usuario puede crear claves temporales distintas para cada una de las tareas que realiza, limitando así el alcance de cualquier posible incidente de seguridad, ya que el compromiso del par de claves asociado a una tarea no afectaría al resto. Por otro lado, el uso de claves temporales y la reducción de las cadenas de delegación permite ocultar la identidad del usuario.

En relación con lo anterior, la aplicación de los solicitantes permite la:

- *Generación de claves temporales.* Es posible crear un número indefinido de claves, tanto RSA como DSA.

- *Delegación de privilegios a claves temporales.* La aplicación permite emitir certificados de autorización SPKI que asocien un subconjunto de los privilegios de los solicitantes a alguna de las claves temporales generadas. De esta forma el usuario puede determinar qué tareas asignar a cada una de dichas claves.

Configuración de autoridades

La aplicación puede almacenar datos relacionados con las distintas autoridades del sistema, los cuales son empleados para generar las solicitudes y encaminarlas hacia la aplicación de la autoridad. En el caso de que la aplicación actúe como un punto de acceso, es posible registrar la información relacionada con el tipo de credenciales que emite cada autoridad y con los parámetros de autorización utilizados para establecer sesiones AMBAR.

Gestión de solicitudes

Al igual que sucedía con la aplicación de las autoridades, aquí también es posible tramitar solicitudes tanto en modo desconectado como en línea. La gestión de las solicitudes se divide en dos operaciones principales:

- *Generación de solicitudes.* Mediante esta opción es posible generar solicitudes tanto de certificación como de reducción. Para ello, tal y como se aprecia en la figura 6.11, el solicitante selecciona la autoridad que deberá emitir el certificado, la clave que firmará la solicitud (es posible crear solicitudes de certificación utilizando alguna de las claves temporales previamente generadas), la credencial que se está solicitando (tanto en el caso de solicitar la pertenencia a un rol como la asignación de un privilegio concreto), la posibilidad de propagar el permiso si éste es obtenido y el periodo durante el cual quiere ejercerse la credencial. En el caso de que la aplicación actúe como punto de acceso, es posible controlar que sólo se soliciten aquellas credenciales que la autoridad seleccionada está autorizada a emitir. Una vez generada la solicitud hay dos posibilidades a la hora de tramitarla: por un lado, en el caso de que la autoridad opere siempre en modo desconectado debido a la sensibilidad de los permisos tratados, será necesario presentar la solicitud utilizando un mecanismo fuera de línea; por otro lado, si la solicitud puede presentarse mediante una conexión AMBAR, la aplicación ofrece la posibilidad de establecer una conexión directa con la autoridad emisora.

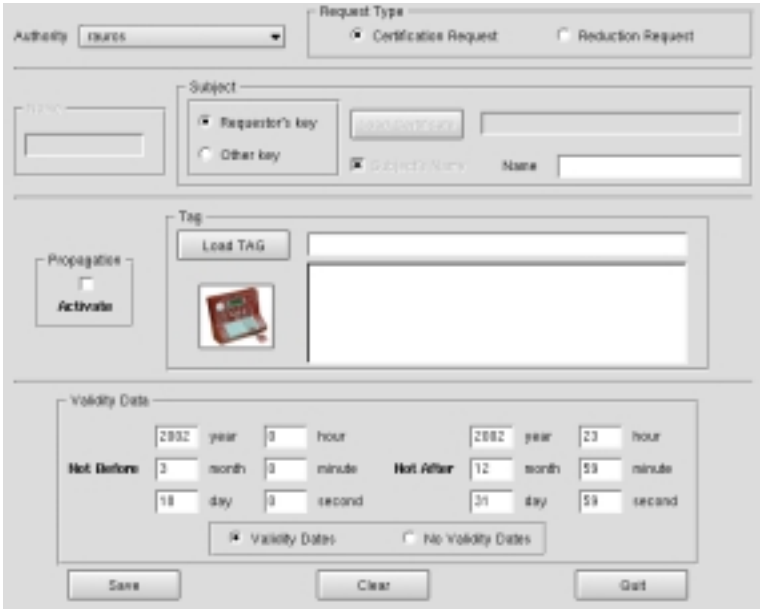

Figura 6.11: Creación de solicitudes

- *Envío de solicitudes.* Para tramitar en línea las peticiones creadas por los solicitantes, la operación de envío de solicitudes permite contactar mediante AMBAR con

cualquiera de las autoridades dadas de alta en la aplicación. Una vez negociada la sesión, los solicitantes pueden enviar cualquier número de solicitudes y de credenciales de apoyo. A continuación, la aplicación mostrará el resultado generado por la autoridad, tanto si se trata de un certificado como de una notificación negativa. En todo momento, es posible monitorizar los mensajes AMBAR que están siendo intercambiados con la autoridad y conocer de esa forma cómo se encapsula la información dentro del protocolo.

6.3.8 Conclusiones

A partir de la descripción realizada en esta sección de la implementación del sistema DCMS, se puede comprobar que su estructura está totalmente condicionada por la definición de la metodología. Esto hace que el conjunto de aplicaciones pueda emplearse de forma ordenada y que los distintos niveles de la metodología tengan una correspondencia con la funcionalidad ofrecida por dichas aplicaciones. Además, el hecho de que se trate de componentes independientes favorece la aplicación selectiva de los procedimientos necesarios y favorece la reutilización de la información generada por cada una de las aplicaciones.

Tanto la aplicación de las autoridades como de los solicitantes ofrecen un amplio abanico de posibilidades, pudiendo además configurarse para que actúen como si de distintos tipos de aplicaciones se tratara. Incorporan todas la funcionalidades criptográficas, de gestión de certificados y de comunicación necesarias para no depender de ninguna otra aplicación externa, excepto de la infraestructura de clave pública tomada como punto de partida.

Por último, dejar constancia de la integración lograda entre el marco AMBAR y el sistema DCMS. Como se ha visto, dicha integración se produce en lo que respecta la configuración de las sesiones, al uso de las primitivas de negociación, envío y recepción, y a la monitorización de los mensajes intercambiados.

A continuación, se mostrará cómo se ha integrado el sistema DCMS en dos entornos de aplicación reales. En primer lugar, se analizará cuáles son las ventajas que puede introducir el uso de certificados de credencial en un escenario tan clásico como es el control de acceso físico a recintos. Posteriormente, se verá cómo se ha diseñado un sistema de suscripción electrónica, o fidelización, que hace uso de un protocolo de pago electrónico desarrollado en el seno del grupo de investigación.

6.4 Integración en un entorno de control de acceso físico

El control de acceso físico implica la provisión de mecanismos que impidan la entrada a determinados recintos, tales como laboratorios de investigación, despachos o almacenes. Este tipo de control es especialmente relevante en aquellos casos en los que los recintos a proteger contienen equipamiento informático, puesto que la manipulación malintencionada de servidores, enrutadores u otros equipos de telecomunicaciones puede llegar a tener un gran impacto sobre algunas comunidades de individuos.

Estamos hablando de un problema clásico, en el cual hay un conjunto de aspectos muy definidos a los que debe aportarse solución. En primer lugar, encontramos el problema de la distribución de claves, es decir, cómo proporcionar las claves apropiadas a los usuarios autorizados que pueden hacer uso de ellas. En relación con esto, las claves pueden ser canceladas o revocadas como respuesta a ciertas situaciones que impliquen una amenaza de seguridad. Por otro lado, es necesario especificar cómo será gestionada la información de los usuarios, es decir, sus datos personales y sus privilegios de acceso. Como ya se ha comentado en apartados anteriores, los usuarios suelen desempeñar ciertos roles dentro del sistema, por lo que resulta muy importante poder reflejar dichos roles en el proceso de gestión de privilegios de acceso.

Tradicionalmente, las principales propuestas se han basado en el uso de bases de datos centralizadas que contenían información acerca de los usuarios autorizados. Por ejemplo, una situación típica es la de un usuario que dispone de datos identificativos, probablemente únicos para cada usuario, que presenta a un dispositivo especial localizado a la entrada de un recinto concreto. En estos entornos clásicos, el dispositivo no conoce qué identificadores son válidos por lo que debe realizar una consulta a la base de datos central para obtener los privilegios del usuario en cuestión. El tipo de identificador puede ser una clave pública, un nombre X.500, un certificado de identidad, datos biométricos o simplemente un nombre de usuario.

Por otro lado, a lo largo de esta tesis se ha demostrado que los certificados de credencial constituyen un mecanismo muy apropiado para codificar información relativa a cualquier tipo de privilegio, lo cual incluye privilegios relacionados con operaciones de control de acceso físico. Además, el uso de este tipo de certificados permite plasmar de forma sencilla las estructuras de roles de un sistema, lo cual simplifica la gestión del mismo.

A lo largo de esta sección, se presentarán dos propuestas [54] que resuelven el problema del control de acceso físico de forma muy distinta. La primera de ellas es el sistema básico que se está empleando en la Universidad de Murcia para controlar el acceso a ciertas instalaciones. La segunda es una modificación que toma como base la propuesta anterior y que proporciona algunas ventajas adicionales desde el punto de vista del no repudio, escalabilidad y la conectividad.

Ambas propuestas hacen uso de un dispositivo especial llamado TICA (Terminal Inteligente de Control de Acceso), el cual ha sido desarrollado completamente en la Universidad de Murcia [136, 160]. Los TICAs contienen lectores de tarjetas inteligentes y son capaces de intercambiar información con un servidor de aplicación a través de una red de comunicaciones ya que están equipados con una tarjeta Ethernet. Además de las operaciones de control de acceso, también son capaces de registrar la hora de entrada y de salida de los trabajadores, así como de realizar operaciones de mantenimiento como el control de iluminación o control de alarmas. Están desarrollados sobre el sistema operativo Linux y su módulo principal de procesamiento está basado en un SBC (Single Board Computer) con una frecuencia de reloj de 133 MHz.

Otro de los elementos comunes a ambas propuestas son las tarjetas inteligentes, las cuales se emplean como repositorios de certificados y como dispositivos habilitados para realizar operaciones criptográficas. En ellas está contenida la información relacionada con

los privilegios de los usuarios, que dependiendo de la propuesta estará constituida por identificadores únicos o por certificados digitales.

En los siguientes apartados se describirán los detalles de las dos propuestas y se mostrará la integración del sistema DCMS así como la aplicación de la metodología a la hora de poner en marcha la propuesta basada en certificados SPKI.

6.4.1 Propuesta centralizada

El sistema actual de control de acceso físico instalado en la universidad sigue un enfoque claramente centralizado. La intención de este análisis es mostrar las carencias asociadas a esta propuesta y cómo pueden ser éstas solventadas empleando otra filosofía en el diseño del sistema. Sin embargo, es importante dejar constancia de que la implementación centralizada actual lleva varios años utilizándose de forma satisfactoria dentro de la comunidad universitaria y en otros escenarios.

La distribución de claves se realiza utilizando identificadores únicos de usuario. Cada persona dispone de su propio identificador (normalmente su número de DNI), el cual se encuentra almacenado en su tarjeta inteligente (recordar que todos los miembros de la Universidad de Murcia disponen de su propia tarjeta inteligente). Por otro lado, existe una base de datos central que contiene una tabla por cada uno de los TICAs que se encuentran instalados. Dichas tablas están compuestas por distintos registros que contienen, entre otros campos, un identificador de usuario y el conjunto de permisos asignados al mismo. Cuando un usuario es autorizado a realizar una acción determinada se introduce un nuevo registro en la tabla asociada al TICA correspondiente. Sin embargo, el sistema no tiene soporte para roles ya que esta solución asigna directamente permisos a usuarios.

Por otro lado, la revocación de claves se implementa de forma bastante sencilla. En el caso de que tengan que anularse los permisos asignados a un usuario, basta con eliminar de la base de datos el registro correspondiente.

En lo que respecta a la verificación de claves, ésta se realiza mediante una consulta remota a la base de datos. Con el fin de garantizar la integridad de los datos intercambiados, se realizan conexiones SSL entre cada uno de los TICAs y los servidores de aplicación que acceden a la base de datos. Cada vez que se instala un nuevo TICA, es necesario generar un par de claves asociadas al dispositivo y solicitar un certificado X.509 a la autoridad de certificación de la universidad, ya que los servidores de aplicación realizan una autenticación de cliente con el fin de verificar que la consulta proviene de un TICA válido. En caso de que se produzca una caída temporal de la red, el dispositivo dispone de una base de datos local que contiene información relacionada con solicitudes de acceso ya tramitadas. Haciendo uso de dicha información local, el TICA puede seguir ofreciendo servicio a aquellos usuarios que ya fueron autorizados en alguna ocasión.

El esquema general de esta propuesta aparece ilustrado en la figura 6.12. Durante la fase de registro inicial, el usuario obtiene su identificador y la correspondiente tarjeta inteligente. En este momento, se puede solicitar también la creación de los registros necesarios para conceder el acceso a determinados recintos. Posteriormente, el usuario introducirá su tarjeta inteligente en el TICA y solicitará una de las operaciones ofrecidas por el dispositivo

(apertura de puertas, control horario del personal, etc.). A continuación, el TICA realizará una consulta a la base de datos central para averiguar si el usuario en cuestión tiene permiso para realizar la acción solicitada. En función de la respuesta, el dispositivo llevará a cabo el servicio o bien notificará al usuario la negativa correspondiente.

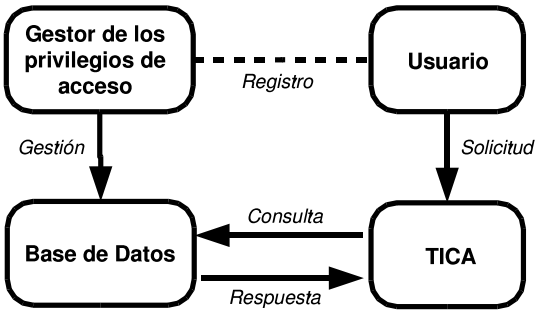

Figura 6.12: Visión general de la propuesta centralizada

6.4.2 Propuesta descentralizada

Aunque la propuesta centralizada se está utilizando de forma satisfactoria y proporciona soluciones a la mayoría de los problemas relacionados con el control de acceso, presenta algunas carencias que pueden ser solventadas siguiendo un enfoque descentralizado basado en el uso de certificados de credencial.

Motivación

El enfoque centralizado requiere una conectividad permanente con la base de datos central. Cuando ésta se rompe, los dispositivos continúan proporcionando servicio basándose en las copias locales de la información de autorización. Por tanto, cualquier modificación posterior de la información contenida en la base de datos será vista sólo por aquellos TICAs que permanecen conectados. En consecuencia, el sistema permanece durante un cierto periodo de tiempo en un estado inconsistente ya que algunos dispositivos rechazarán solicitudes que sin embargo otros aceptarán.

De hecho, en algunos entornos no resulta sencillo disponer de conectividad a la red de comunicaciones, lo cual impide poder utilizar un enfoque de este tipo. Sería aconsejable que el sistema de control de acceso fuera realmente distribuido, no por el hecho de estar basado en dispositivos distribuidos geográficamente sino por la posibilidad de ejercer sus funciones sin depender de un punto central. En la propuesta descentralizada que aquí se presenta los terminales pueden operar en modo desconectado y pueden determinar si un usuario está autorizado a realizar la acción solicitada sin necesidad de consultar a ninguna entidad externa.

Por otro lado, si el escenario en el cual se desarrolla en sistema de control de acceso está constituido por una comunidad de usuarios extensa, la tarea de gestionar cada TICA,

la lista de usuarios autorizados o la copia local de la información de autorización puede resultar muy compleja. Una solución más acertada es definir grupos de usuarios a los cuales asignar conjuntos de permisos en lugar de gestionar cada usuario de forma individual. No obstante, realizar dicha definición haciendo uso de una base de datos central presenta los mismos problemas derivados de la falta de conectividad que aparecían en el caso de las autorizaciones individuales. Por tanto, el mecanismo de definición de grupos o roles debe tener también un enfoque distribuido.

Por último, la propuesta centralizada carece de mecanismos robustos de no repudio de solicitante. Es cierto que tanto la base de datos como los TICAs realizan apuntes de las acciones que han sido solicitadas por los usuarios. Sin embargo, dichos apuntes no constituyen un mecanismo robusto de cara a ser utilizados como prueba de no repudio. La información registrada es susceptible de ser modificada por cualquier intruso capaz de acceder a parte de la información almacenada en la base de datos. Aunque es posible utilizar algunas soluciones basadas en el cifrado de datos para almacenar información confidencial en sistemas no confiables, lo realmente necesario es poder disponer de información generada por el solicitante, y no por los TICAs o la propia base de datos, que pueda ser utilizada como una evidencia irrefutable de las peticiones realizadas. Más concretamente, se está haciendo referencia al uso de solicitudes de servicio firmadas digitalmente por los usuarios, las cuales podrán ser empleadas, junto con las decisiones de autorización, para adoptar las medidas pertinentes tras un incidente de seguridad.

Diseño de la propuesta

La propuesta descentralizada está completamente basada en el uso del sistema DCMS, lo que implica que la gestión de las autorizaciones se realice según el esquema RBAC (ver sección 4.2.3) y el mecanismo de delegación de privilegios.

Por un lado, el esquema RBAC permitirá separar la gestión de la pertenencia de los usuarios a roles de la asignación de privilegios concretos a dichos roles, lo cual simplifica enormemente la gestión de las autorizaciones. La pertenencia se implementará mediante la emisión de certificados SPKI de identidad mientras que la asignación de privilegios a los roles se realizará mediante certificados SPKI de atributo. Los certificados de pertenencia se almacenarán siempre en la tarjeta inteligente del usuario asociado, el cual dispondrá normalmente de un certificado de identidad X.509 emitido por la PKI presentada en el capítulo 3. Los certificados de atributo podrán ser almacenados tanto en las tarjetas de los usuarios como en los propios TICAs. Esto es debido al hecho de que los permisos asociados a un rol suelen cambiar menos frecuentemente que el conjunto de personas que lo componen, por lo que la opción de almacenarlo en los TICAs no presentará normalmente problemas de actualización y liberará a los usuarios de agotar el escaso espacio de almacenamiento proporcionado por las tarjetas inteligentes.

Por otro lado, la delegación de privilegios permite que los TICAs asignen la responsabilidad de la gestión de los mismos a entidades externas que actuarán como autoridades. De esta forma, el TICA no es sólo el punto de cumplimiento de la política de control de acceso sino también el inicio de la cadena de autorización. El dispositivo es capaz de tomar

decisiones de autorización analizando tanto los certificados de credencial que mantiene almacenados como los presentados por los usuarios. El esquema general utilizado, mostrado en la figura 6.13, se corresponde con el modelo de gestión propio del sistema DCMS.

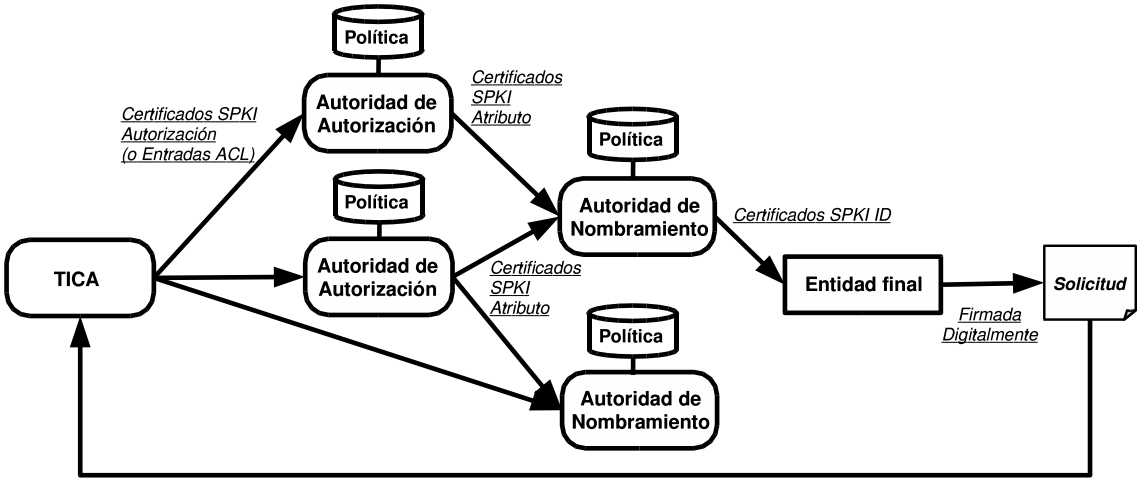

Figura 6.13: Arquitectura del sistema de control de acceso descentralizado

Como se puede apreciar, el usuario presenta al TICA solicitudes de acceso firmadas digitalmente mediante su clave privada. A continuación, el TICA intenta construir una cadena de certificados que partiendo de él mismo sea capaz de llegar hasta la clave pública del usuario pasando por las distintas autoridades del sistema. En el caso de que logre hallar una prueba de autorización, el dispositivo lleva a cabo la acción correspondiente y almacena dicha prueba con el propósito de registrar evidencias que puedan ser utilizadas en caso de incidencia.

6.4.3 Implementación de la propuesta distribuida

La aplicación de control de acceso del TICA está también basada en la arquitectura CDSA. En concreto, el grupo de investigación ANTS desarrolló una librería multimódulo [173] que actuaba como proveedor de servicios criptográficos (CSP) y como módulo de almacenamiento de datos (DL).

El módulo CSP proporciona soporte para el uso de las tarjetas inteligentes y está encargado de la realización de las operaciones de firma digital que implican el uso de la clave privada contenida en la tarjeta. La gran ventaja de este enfoque es que la aplicación del TICA accede a la tarjeta de forma transparente a través de la API proporcionada por CSSM, sin que tenga que conocer ninguna interfaz de programación específica desarrollada para tal efecto.

Por otro lado, dado que los certificados de credencial SPKI se encuentran almacenados en la tarjeta, era necesario desarrollar un módulo DL que ofreciera la posibilidad de recuperar esos datos a través de la API CSSM.

6.4.4 Aplicación de la metodología e integración con DCMS

Para mostrar cómo es posible hacer uso de la metodología y de DCMS para poner en marcha el sistema de control de acceso físico anteriormente descrito, se planteará aquí un escenario concreto formado por un conjunto de TICAs, roles y usuarios. A continuación, se aplicará la metodología para determinar tanto las políticas de control de acceso del sistema como las entidades encargadas de su cumplimiento. Una vez finalizados los procedimientos, el sistema DCMS podrá ser utilizado para generar los certificados en base a los cuales tomarán sus decisiones los TICAs del sistema.

Escenario a gestionar

El escenario de partida está formado por cuatro TICAs distintos, los cuales pueden realizar operaciones relacionadas con la apertura de puertas, control horario, control de iluminación y gestión de alarmas. Todos los TICAs están localizados en la Facultad de Informática, concretamente en la puerta principal, puerta posterior, puerta del laboratorio de investigación del departamento DITEC y puerta del laboratorio de investigación del departamento DIIC.

Por otro lado, los usuarios se agrupan en siete roles distintos. Con el fin de acotar la extensión de este apartado, supondremos que la jerarquía formada por dichos roles ya ha sido definida mediante la aplicación de los procedimientos correspondientes al bloque NMS de la metodología. Dicho bloque de procedimientos será utilizado aquí sólo para gestionar la pertenencia de los usuarios a los roles. La figura 6.14 muestra la jerarquía concreta de roles.

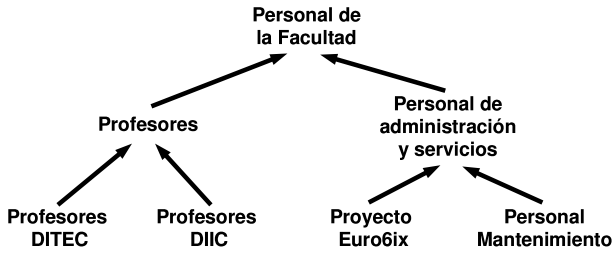

Figura 6.14: Roles de la jerarquía

Aplicación de los procedimientos de nivel 0

A continuación se desglosa cómo se han aplicado los procedimientos contenidos en este nivel:

- *Especificación de los recursos a proteger y sus operaciones.* Los recursos que se desea gestionar son los propios TICAs del sistema. Para especificarlos se llevarán a cabo los siguientes pasos:

  - *Establecimiento del esquema de identificación de recursos.* Para identificar a los dispositivos se utilizará una notación basada en la concatenación del nombre de la facultad en la cual se encuentra el TICA y de la localización física del mismo. Por ejemplo, el dispositivo situado en la puerta principal se denotará con la expresión `(informatica/principal)`. El resto de dispositivos tienen los identificadores `(informatica/posterior)`, `(informatica/sala-diic)` e `(informatica/sala-ditec)`. Esta notación nos permite hacer referencia a todos los TICAs de una misma facultad mediante el uso de expresiones del tipo `(* prefix informatica/)`.
  - *Establecimiento de las operaciones sobre los recursos.* En este caso concreto podemos identificar cinco operaciones a controlar. La apertura de las puertas se representa con la expresión `(open-door (* range time (hh:mm:ss) (hh:mm:ss)))`, la cual permite especificar el intervalo horario durante el cual podrá realizarse la apertura de la puerta controlada por el TICA. El control horario de los trabajadores tiene asociadas dos operaciones `(start-work)` y `(finish-work)`, utilizadas respectivamente para fichar al inicio y al final de la jornada laboral. Por último, las operaciones de mantenimiento son representadas mediante las expresiones `(lighting-control)` y `(alarm-management)`.
  - *Publicación de la especificación.* La representación de los recursos y sus operaciones se hará pública mediante la implementación de una aplicación generadora de tags, tal y como se comentó en la sección 6.3.2. Dichos tags contendrán siempre la cadena *dkronos*, la cual hace referencia al tipo de privilegios que se están gestionando (el nombre *dkronos* está derivado a partir del nombre KRONOS con el cual se denominó al proyecto mediante el cual se construyeron los TICAs).

- *Determinación de los controladores.* En este caso concreto, los controladores son los propios TICAs ya que serán los encargados de verificar las solicitudes presentadas por los usuarios.

- *Determinación de las autoridades.* El siguiente paso es la delegación de la gestión de privilegios por parte de cada uno de los TICAs. Para ello hay que determinar las entidades que actuarán como autoridades y generar la correspondiente lista de control de acceso o certificado de autorización, generación que puede realizarse respectivamente con la aplicación generadora de políticas (ver sección 6.3.5) o con la aplicación de las autoridades (ver sección 6.3.6). En este caso concreto, la determinación de autoridades se ha realizado de la siguiente forma: los TICAs de la puerta principal y posterior delegan la gestión de todas las operaciones en una autoridad que llamaremos *AA-Facultad*; el TICA del laboratorio de DITEC delega en *AA-DITEC* la gestión de la operación de apertura de puertas ; por último el TICA del laboratorio de DIIC delega también la gestión de la apertura en la autoridad *AA-DIIC*.

En el caso de que la delegación se realizara mediante la construcción de listas de control de acceso, estas tendrían una estructura similar a la especificada en la figura 6.15.

```
(acl
  (subject AA − Facultad)
  (tag (dkronos
         (* set (informatica/principal) (informatica/posterior))
         (*)))
)
(acl
  (subject AA − DITEC)
  (tag (dkronos (informatica/sala-ditec) (open-door (*))))
)
(acl
  (subject AA − DIIC)
  (tag (dkronos (informatica/sala-diic) (open-door (*))))
)
```

Figura 6.15: Delegación de los controladores mediante ACLs

Aplicación de los procedimientos del bloque AMS

El siguiente paso es la determinación de la política de autorización por parte de las autoridades derivadas de la aplicación de los procedimientos de nivel 0. Para ello se emplean los siguientes niveles del bloque AMS:

- *Nivel 1. Identificación de las relaciones entre las operaciones.* En el caso concreto del entorno que se está diseñando, las cinco operaciones posibles pueden agruparse en tres grupos distintos: apertura de puertas, control horario y mantenimiento. Además, en el caso de la operación de apertura hay que determinar las franjas horarias más comunes a emplear, por ejemplo la correspondiente con el horario de apertura de la facultad.

- *Nivel 2. Asignación de permisos a entidades receptoras.* Cada una de las autoridades seguirá un criterio de asignación distinto en función de los permisos que esté gestionando. En este caso concreto, se realizarán las siguientes asignaciones (los periodos de validez se omiten por simplicidad):

  - *AA-Facultad.* Establecerá la siguiente política de autorización: el personal de la facultad podrá realizar las operaciones de control horario en cualquiera de los TICAs; el personal de mantenimiento podrá solicitar las operaciones de control de iluminación y de gestión de alarmas tanto mediante el dispositivo de la puerta principal como mediante el de la puerta posterior; por último, el personal de la facultad podrá abrir la puerta principal a cualquier hora. Las s-expresiones que codifican estas condiciones se muestran en la figura 6.16
  - *AA-DITEC.* La política de autorización de esta autoridad está compuesta de una única autorización, la que permite a los profesores de DITEC abrir la puerta de su laboratorio de investigación a cualquier hora.

– *AA-DIIC.* Esta autoridad autoriza tanto a los profesores de DIIC como a los miembros del proyecto Euro6ix a acceder a su laboratorio a cualquier hora del día.

```
(cert-request
  (issuer AA − Facultad)
  (subject Personal)
  (tag (dkronos (*) (* set (start-work) (finish-work))))
)
(cert-request
  (issuer AA − Facultad)
  (subject Personal)
  (tag (dkronos (informatica/principal) (open-door)))
)
(cert-request
  (issuer AA − Facultad)
  (subject Mantenimiento)
  (tag (dkronos
         (* set (informatica/principal) (informatica/posterior))
         (* set (lighting-control) (alarm-management))))
)
```

Figura 6.16: S-expresiones de la autoridad AA-Facultad

- *Nivel 3. Determinación de los solicitantes y periodos de solicitud.* Al tratarse de privilegios asignados a roles, es necesario especificar qué entidad, o conjunto de entidades, estará autorizada a solicitar los certificados asociados a los privilegios. En este caso, se supondrá que todas las autoridades consideran que los solicitantes válidos son las autoridades de nombramiento encargadas de gestionar la pertenencia a los roles autorizados. Se permitirá además que dichas autoridades puedan delegar dicha posibilidad de solicitud a otras entidades.

- *Nivel 4. Modos de acceso a la autoridad.* En este caso supondremos que todas las autoridades han configurado su aplicación de forma que acepte conexiones AMBAR procedentes de cualquier origen. Además, las sesiones deben ser identificadas y deben utilizar el método de distribución *push-asserts*.

Aplicación de los procedimientos del bloque NMS

Por último, deben aplicarse los procedimientos del bloque NMS con el fin de establecer las políticas de las distintas autoridades de nombramiento del sistema. Se supone que dentro del sistema se han definido tres autoridades de nombramiento distintas: *NA-Facultad* se encarga de gestionar la jerarquía de roles mostrada en la figura 6.14 y la pertenencia al grupo *Personal de Mantenimiento*; por otro lado, *NA-DITEC* tiene la responsabilidad de

especificar y emitir los certificados de pertenencia al rol *Profesores DITEC*; finalmente, la autoridad *NA-DIIC* gestiona la pertenencia tanto al rol *Profesores DIIC* como al rol *Proyecto Euro6ix*. Todas ellas llevarán a cabo los procedimientos de los siguientes niveles para especificar sus políticas:

- *Nivel 1. Identificación del conjunto de elementos.* En primer lugar, es necesario delimitar el conjunto de entidades que se encuentra dentro de la influencia de cada autoridad. Para el caso de *NA-Facultad* serán todas aquellas personas que pertenecen al personal de administración y servicios, como los administradores del centro de cálculo, personal de secretaría y conserjes. En el caso de las otras dos autoridades dicho conjunto abarca tanto a los profesores de cada departamento como al conjunto de becarios, contratados con cargo a proyectos de investigación, profesores visitantes y alumnos de proyecto fin de carrera.

- *Nivel 2. Determinación de la pertenencia.* Durante esta etapa, las autoridades deben decidir qué usuarios pertenecerán a cada uno de los roles que gestionan. Para este ejemplo, se supondrá que el rol *Personal de Mantenimiento* estará formado por los conserjes de la facultad, y que los roles de los profesores estarán constituidos exclusivamente por los profesores de cada departamento. Por último, formarán parte del *Proyecto Euro6ix* aquellos contratados y becarios que se encuentren implicados en el proyecto. Todas estas decisiones podrán ser plasmadas en s-expresiones haciendo uso de la aplicación de gestión de identificadores (ver sección 6.3.4).

- *Nivel 3. Determinación de los solicitantes y periodos de solicitud.* Se supondrá que las tres autoridades han decidido que el conjunto de solicitantes válidos esté formado por las entidades pertenecientes a cada uno de los roles, es decir, los usuarios podrán solicitar personalmente sus certificados. En consecuencia, durante la construcción de la política mediante la aplicación generadora, no será necesario especificar el conjunto de solicitantes válidos por ser equivalente al conjunto de receptores válidos especificados en las s-expresiones de tipo *cert-request*.

- *Nivel 4. Modos de acceso a la autoridad.* En este caso supondremos que todas las autoridades han configurado su aplicación para que sólo acepte conexiones AMBAR que procedan de puntos de acceso. Además, las sesiones deben ser identificadas y deben utilizar el método de distribución. *push-asserts*.

6.4.5 Conclusiones obtenidas

En vista de las características de las dos propuestas presentadas, y del uso que se ha realizado tanto de la metodología como del sistema DCMS a la hora de poner en marcha la propuesta descentralizada, es posible extraer varias conclusiones:

- La propuesta basada en el uso de certificados de credencial genera datos que pueden ser utilizados como evidencias a la hora de garantizar el no repudio del solicitante.

- La escalabilidad del sistema se ve mejorada mediante la aplicación del modelo RBAC y el uso de certificados SPKI, ya que es posible dividir las responsabilidades de gestión de forma natural entre las distintas entidades del sistema.

- La versión distribuida es capaz de proporcionar servicio en aquellos entornos en los cuales la conectividad a una red de comunicaciones no es posible.

- La aplicación de la metodología permite poner en marcha el sistema de forma estructurada y clara. Además, la aplicación de los procedimientos se puede realizar de forma inmediata mediante el uso de las aplicaciones que forman parte del sistema DCMS.

En resumen, el uso de la infraestructura de autorización amplía las posibilidades de los sistemas de control de acceso físico centralizados, todo ello desde un punto de vista estructurado y metodológico que simplifica además el proceso de diseño del sistema.

6.5 Integración en un entorno de suscripción electrónica

Además del entorno analizado en la sección anterior, se creyó conveniente mostrar cómo la infraestructura de autorización puede integrarse también en escenarios más enfocados al comercio electrónico, como es el caso de la fidelización de clientes.

La fidelización de clientes es uno de los principales objetivos de la mayoría de las empresas de cara a asegurar una determinada cuota de mercado. Las empresas buscan consolidar en el tiempo sus comunidades de clientes, lo cual suele repercutir positivamente en ambas partes. Para ello, se han diseñado multitud de fórmulas que tienen como objetivo poner al alcance de los clientes ventajas adicionales respecto al cliente eventual, casi siempre enfocadas al plano económico (descuentos especiales, ofertas, facilidades de pago, etc.). Uno de los mecanismos que más se ha empleado para este tipo de propósitos es la posibilidad de pagar una cuota de suscripción que asocie al cliente a un determinado grupo beneficiario. Dependiendo de la cuota, el cliente disfrutará de un conjunto de ventajas adicionales, las cuales serán mejores cuanto más alta sea la cuota de suscripción.

En esta sección, se expondrá una propuesta que proporciona los mecanismos necesarios tanto para realizar el pago de la cuota de suscripción como para hacer uso de la misma a la hora de comprar de forma electrónica a través de una red de comunicaciones. El sistema está basado en el uso del protocolo SPEED [174], desarrollado en el marco del proyecto PISCIS. SPEED proporciona los medios necesarios para realizar de forma segura pagos basados en el uso del monedero electrónico, distribuir de forma electrónica el producto solicitado, generar toda la información necesaria para resolver posibles disputas que pudieran surgir en el futuro e intercambiar información relativa a atributos o credenciales de las entidades participantes. Como se verá, el uso de dicho protocolo, a través de su integración con la infraestructura de clave pública y el sistema DCMS, permite diseñar un sistema de suscripción electrónica especialmente enfocado a sistemas distribuidos en los cuales no es apropiado realizar una gestión centralizada de las suscripciones.

6.5.1 El protocolo SPEED

En los últimos años, la comunidad científica ha ido tomando conciencia de la necesidad de diseñar e implementar nuevas formas de pago adaptadas al comercio electrónico que hagan un buen uso de la tecnología existente y proporcionen al usuario un cierto grado de percepción de seguridad. En general, cada uno de estos sistemas propuestos intenta satisfacer las necesidades del entorno en el cual está definido, y por tanto no podemos considerar que haya un sistema válido para cualquier entorno. Algunas de estas propuestas [142] han demostrado ser lo suficientemente seguras y flexibles, si bien no han alcanzado un alto grado de adopción en mercados reales.

Durante el desarrollo del proyecto PISCIS [53] se identificó el conjunto de características de seguridad que debería reunir un protocolo de pago de cara a poder ser utilizado en un escenario real de comercio electrónico. Entre dichos requisitos se encontraba la siguiente lista:

- El sistema de pagos debe ser capaz de ofrecer medios para negociar el precio de los productos o servicios ofrecidos por los comerciantes.
- La entrega electrónica de los bienes adquiridos debe formar parte del sistema.
- El protocolo debe estar basado en estándares de seguridad reconocidos. Este requisito incrementa la sensación de seguridad percibido por los usuarios, y garantiza un diseño basado en propuestas debatidas y probadas por la comunidad científica.
- Se debe disponer de elementos de arbitraje capaces de ejercer como mediadores y como entidades de confianza a la hora de mediar en conflictos y situaciones excepcionales.

Aunque algunos de estos requisitos ya habían sido satisfechos por algunas de las propuestas ya existentes, con el fin de proporcionar una respuesta común a todos ellos dentro del marco de trabajo del Proyecto PISCIS, se definió un nuevo sistema de pago llamado SPEED (Smartcard-based Payment with Encrypted Electronic Delivery), el cual proporciona, como su propio nombre indica, un sistema de pago basado en monedero electrónico para tarjeta inteligente con entrega cifrada de bienes. Aunque la información en detalle del protocolo puede encontrarse en [174], en esta sección haremos una breve descripción de sus características principales, participantes y modelo de compra.

Visión general

Una transacción SPEED transfiere bienes electrónicos desde un vendedor a un cliente, debitando el monedero electrónico del cliente e incrementando el saldo de la cuenta del vendedor por el valor de producto. El diseño de SPEED consiste en una serie de fases que incluyen la negociación del precio, la entrega del producto y su pago. Además, hay dos modos posibles de operación: el modo normal incluye la capacidad de negociación del precio del producto, y ha sido diseñado para proporcionar el mayor número de características de

seguridad (como por ejemplo la prevención de ataques de denegación de servicio y la autenticación completa de las partes participantes antes del suministro del producto); el modo rápido de operación está compuesto por un número menor de mensajes que el modo normal, y está pensado para la venta de bienes de menor tamaño o escenarios con menores requisitos de seguridad.

Su diseño se basa en el uso de estándares como ASN.1 [105] para la especificación de la estructura de los mensajes, PKCS#7 [118] como formato criptográfico para el intercambio de información protegida, certificados X.509v3 [99] para la identificación de los participantes en el escenario de compra y WG10 [39] como sistema estándar de monedero electrónico.

Participantes

El modelo de negocio de SPEED está compuesto por tres entidades principales: el cliente, el comerciante y el intermediario (broker). El broker gestiona las cuentas de los comerciantes (y opcionalmente las de los clientes) y mantiene el conjunto de módulos de seguridad que realizan las operaciones de decremento sobre el monedero electrónico del cliente. Esta entidad no interviene hasta la fase de pago, una vez que el cliente envía la solicitud de transacción.

En una primera instancia, el cliente y el vendedor acuerdan el producto a comprar y su precio, lo cual puede ser llevado a cabo después de una fase de negociación de ofertas que es opcional. El producto se transmite al cliente cifrado con una clave simétrica, que sólo le es proporcionada una vez que el pago correspondiente se ha materializado. Cuando dicho pago se ha realizado, tanto el cliente como el vendedor obtienen una prueba del resultado de la transacción (denominada recibo).

Modelo de compra

La figura 6.17 muestra un esquema global de las comunicaciones que componen una secuencia de compra de SPEED en el modo normal de operación. Los mensajes 1, 2 y 3, intercambiados entre el cliente y el comerciante, constituyen la fase de negociación del producto. El mensaje 4 contiene el producto cifrado con una clave simétrica generada aleatoriamente por el comerciante, y que será proporcionada al cliente una vez que el pago se haya realizado (mensajes 5 y 6). El broker y el cliente intercambian una serie de mensajes adicionales destinados a realizar el proceso de decremento del monedero electrónico (en la figura estos mensajes están representados por la línea punteada). Todas las comunicaciones están protegidas frente a ataques de entidades externas haciendo uso de criptografía simétrica principalmente.

En relación con el mecanismo de suscripción electrónica, es importante analizar en detalle el formato del primer mensaje del protocolo (*NegotiationRequest*), el cual está encuadrado dentro de la fase de negociación del precio del producto. Se trata de un mensaje firmado digitalmente que el cliente envía al vendedor para preguntar o proponer el precio de un determinado producto. Su estructura es la siguiente:

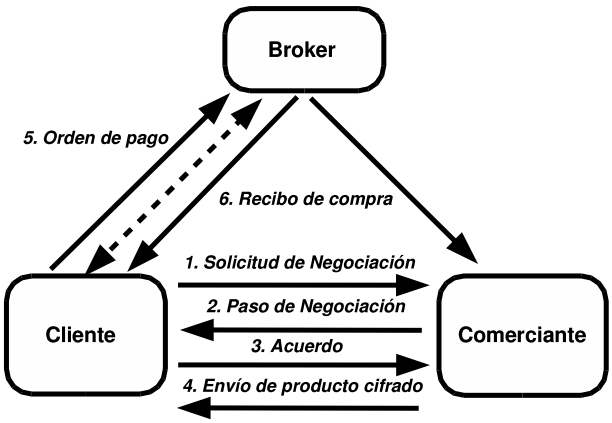

Figura 6.17: Modelo de compra de SPEED

$$1 \quad \mathrm{C} \Rightarrow \mathrm{V} \quad \mathit{NegotiationRequest} \quad \{\{NID, SeqN, ProductID, [Price], VendorID, EnKey, SignKey, Flag, Credential\}_{C^{-1}}\}_V$$

Entre todos los campos que forman este mensaje, son de especial interés para esta sección los relacionados con la descripción del producto (*ProductID*), el precio (*Price*) y las credenciales (*Credential*):

- *ProductID (Product Identifier).* Se trata de una cadena de octetos que representa al producto a comprar. Es totalmente dependiente del entorno de aplicación en el cual se encuentre ubicado el protocolo, y en este caso concreto servirá tanto para designar a los certificados de pertenencia como a los productos finales a comprar.

- *Price.* Es el precio que el cliente está dispuesto a pagar por el producto. Se trata de un elemento opcional ya que es posible que el cliente no conozca de antemano el precio.

- *Credential.* Durante el proceso de diseño del protocolo SPEED ya se consideró la posibilidad de proporcionar un soporte especial que permitiera realizar la transmisión de información relativa a credenciales como parte del protocolo. El propósito genérico de este tipo de información era la modificación de la estrategia de negociación del proveedor, un cambio de estrategia que implicara ciertas ventajas para el cliente. Dicha información se incluye dentro del campo *Credential* y su formato concreto es totalmente transparente para el protocolo ya que éste sólo se encarga de transmitir la información sin interpretar su estructura.

Tal y como se verá más adelante, el uso de estos campos del protocolo permitirá diseñar tanto el escenario de solicitud de suscripción como el de disfrute de la misma. El resto de los campos así como de los mensajes del protocolo son totalmente independientes del mecanismo aquí diseñado.

6.5.2 Integración de la PKI

Cada participante de SPEED (clientes, comerciantes y broker) poseerá una clave privada RSA almacenada en su tarjeta inteligente y un certificado X.509 emitido por la infraestructura de clave pública presentada en el capítulo 3. De hecho, SPEED asume la existencia de relaciones de confianza entre las entidades participantes. Los brokers son considerados las entidades de mayor confianza, seguidos de los comerciantes y por último de los clientes (en los cuales podría no tenerse ningún tipo de confianza). Los brokers juegan el rol de participar como entidades intermediarias y los comerciantes poseen relaciones a largo plazo con los brokers de la misma forma que lo harían con un banco. La reputación del broker dentro del sistema es un punto importante, ya que resulta vital que asuman su papel según lo establecido con el fin de no perder la confianza del resto de las entidades participantes.

Por tanto, la relación con la PKI tiene dos puntos de unión claramente diferenciados. Por un lado, todas las entidades participantes deben obtener un certificado digital X.509 que les permita establecer comunicaciones confidenciales y autenticadas entre ellos. La identidad digital contenida en dichos certificados será vital a la hora de resolver posibles disputas que pudieran surgir en el futuro. Además, en el caso de los clientes, este proceso de certificación implica el uso de tarjetas inteligentes como elementos tanto contenedores de información sensible como capaces de realizar operaciones criptográficas.

Por otro lado, la infraestructura de clave pública proporciona los mecanismos de validación en línea del estado de los certificados implicados en las transacciones. Estos servicios son de vital importancia a la hora de evitar posibles fraudes derivados del uso malintencionado de información criptográfica ajena.

6.5.3 Implementación de la suscripción electrónica mediante certificados de credencial

El escenario de suscripción electrónica realizado está basado en la existencia de varias categorías de suscripción, gestionadas posiblemente por entidades distintas, y varios proveedores de servicios que ofrecen ventajas económicas a los suscriptores, las cuales variarán dependiendo del tipo de suscripción. Tal y como se muestra en la figura 6.18, los distintos proveedores de contenidos y servicios mantienen una relación de colaboración con los diferentes gestores de suscripciones. Los términos de dicha colaboración especificarán qué tipo de ventajas aplicará el proveedor a aquellos miembros de alguno de los grupos que controla el gestor (descuentos en algunos de sus productos, facilidades de pago, regalos, etc.).

Los usuarios finales tienen dos puntos de conexión con el sistema de suscripción. Por un lado deberán pagar la cuota correspondiente y obtener el justificante de suscripción. Por otro lado, deberán transmitir dicho justificante junto con cada solicitud de compra realizada a alguno de los proveedores relacionados con el grupo de suscripción.

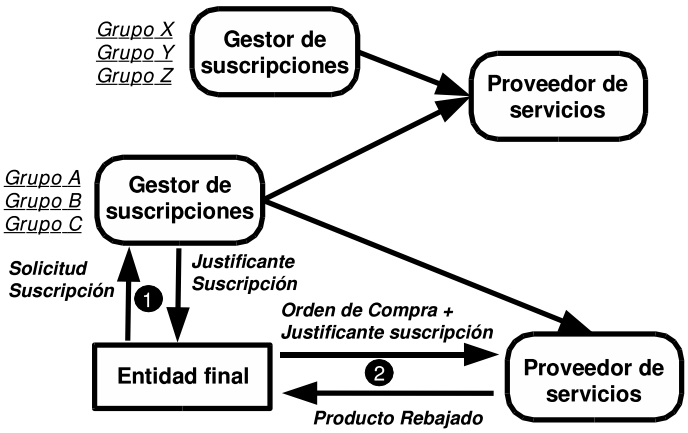

Figura 6.18: Modelo de suscripción electrónica

Solicitud de suscripción

La solicitud de suscripción se realiza a través del protocolo SPEED. Dado que dicho protocolo fue diseñado especialmente para el pago de productos digitales, es posible especificar cuál es el grupo concreto al cual se desea pertenecer, realizar el pago correspondiente y obtener el justificante de suscripción.

En relación con lo visto en el apartado 6.5.1, la especificación del grupo está contenida en el campo *ProductID* del mensaje *NegotiationRequest*, el cual contendrá una s-expresión de tipo *issuer* [68] con la clave pública del gestor de suscriptores y el identificador del grupo. La suscripción, cuyo precio está especificado en el campo *Price*, podría estar sujeta al cumplimiento de ciertas condiciones de autorización (por ejemplo, ser mayor de edad, tener crédito disponible o cualquier otro criterio dependiente del entorno). En dicho caso, las evidencias necesarias para demostrar la posibilidad de ingreso en el grupo podrían transmitirse en el campo *Credential*.

Una vez que el broker transmite el recibo, el usuario final puede descifrar el justificante de suscripción, el cual recibió como parte del cuarto mensaje del protocolo. Dicho justificante estará representado mediante un certificado SPKI de identidad, el cual ligará la clave pública del usuario con el grupo de suscripción, todo ello durante el periodo de tiempo en el cual sea efectiva dicha asociación.

El uso de certificados digitales permite descentralizar el sistema puesto que no es necesario mantener almacenadas de forma central todas las relaciones existentes entre los usuarios del sistema y los grupos. En contraposición, la información de suscripción puede ser transmitida por los propios usuarios a la hora de realizar sus compras y será considerada válida por aquellas entidades que tienen una relación de confianza con la entidad emisora.

Presentación de justificantes

Tras la adquisición del justificante, el usuario final puede proceder a la compra de productos de alguno de los proveedores de servicios con los cuales tenga acuerdos el gestor de

suscripciones. La descripción de dichos productos se incluirá en el campo *ProductID* del primer mensaje.

Por otro lado, haciendo uso del campo *Credential* del mensaje *NegotiationRequest*, el cliente puede transmitir al proveedor el justificante de suscripción, es decir, el certificado de identidad SPKI que le asocia a un determinado grupo. Una vez que el proveedor recibe el certificado, verifica su estado y determina la reducción en el precio del producto que ha sido solicitado. Dicha reducción depende del acuerdo establecido entre el proveedor y la entidad gestora de las suscripciones, el cual especificará el conjunto de productos afectados, los porcentajes de descuento u otras medidas favorecedoras, y el periodo de tiempo durante el cual permanecerá en vigor. En concreto, en el prototipo desarrollado, los proveedores de contenido actúan como autoridades de autorización, emitiendo certificados SPKI de atributo que especifican los acuerdos de los cuales se pueden beneficiar aquellos usuarios pertenecientes al grupo de suscriptores al que hace referencia el certificado. Dichos certificados pueden emplearse como prueba de los acuerdos alcanzados entre las entidades del sistema, lo cual es especialmente útil a la hora de decidir el grupo de suscriptores al cual se desea pertenecer.

6.5.4 Conclusiones obtenidas

Uno de los principales criterios de diseño del protocolo SPEED fue la posibilidad de ofrecer un mecanismo de negociación del precio de los productos que formara parte del propio protocolo, lo cual permitiría emplearlo en escenarios basados en entidades mediadoras y gestionar comunidades con distintas prioridades. Respecto a este último aspecto, el uso de certificados de credencial ha permitido introducir de forma estructurada un servicio de fidelización que aprovecha parte de la funcionalidad ofrecida por el protocolo. En consecuencia, el uso de la PKI y de la infraestructura de autorización aportan los mecanismos necesarios de confidencialidad, integridad, no repudio, autenticación y autorización.

6.6 Evaluación de los componentes de la infraestructura de autorización

Con el fin de evaluar el rendimiento proporcionado por los dos componentes fundamentales de la infraestructura, la implementación del marco AMBAR y el sistema DCMS, se realizó un conjunto de pruebas destinadas a obtener conclusiones respecto a los siguientes tres aspectos:

- Comparar el rendimiento ofrecido por el protocolo AMBAR frente a uno de los protocolos de seguridad más utilizados, el protocolo SSL. Dicho análisis abarca tanto la fase de negociación o *handshake* como el envío posterior de datos.

- Averiguar cuál es la sobrecarga que introduce la arquitectura CDSA frente a implementaciones que hagan uso de la funcionalidad criptográfica de forma directa.

- Averiguar cuál es el peso tanto de las comunicaciones AMBAR como del procesamiento DCMS a la hora de tramitar las solicitudes de certificación o reducción, es decir, determinar la contribución de cada elemento al tiempo total.

En los siguiente apartados se presenta tanto el entorno en el cual se realizaron las pruebas como los resultados obtenidos para cada uno de los objetivos fijados.

6.6.1 Entorno de evaluación

Los elementos participantes en las pruebas son un cliente que solicita el acceso a un recurso y el controlador correspondiente. Salvo en las pruebas que involucran al sistema DCMS, las implementaciones de tanto el solicitante como el controlador son básicas, es decir, incluyen exclusivamente la funcionalidad necesaria para establecer canales de comunicación seguros e intercambiar información de autorización.

Las pruebas se han desarrollado en una red de área local Ethernet (10 Mbps) conmutada. Los ordenadores donde se ejecutan los procesos cliente y servidor son Intel Pentium III con 192 MB de memoria principal. El sistema operativo empleado fue Linux (Kernel 2.4.2-2) y todas las aplicaciones fueron compiladas mediante GCC 2.96. Por último, el software criptográfico utilizado fue la implementación de Intel de la arquitectura CDSA (versión 3.14) y las librerías OpenSSL (versión 0.9.6).

Por otro lado, los parámetros relacionados con la configuración de sesiones AMBAR y con las características de la información de autorización intercambiada pueden observarse en la tabla 6.1.

Parámetro	**Valor**
Cifrador simétrico	RC2 (128 bits)
Cifrador asimétrico	RSA (1024 bits)
Algoritmo firma digital	DSA (1024 bits)
Tamaño de los certificados SPKI	1350 bytes
Tamaño de los certificados X.509	700 bytes
Tamaño de la solicitud	1700 bytes
Tamaño de la política	600 bytes
Método de distribución	PUSH-ASSERTS
Modo de conexión	Identificado

Tabla 6.1: Parámetros de la evaluación

6.6.2 Evaluación de la fase de negociación

La primera prueba realizada fue la evaluación del tiempo de ejecución asociado a la fase de negociación de sesiones AMBAR. Para ello, se realizó una comparativa respecto a la fase *handshake* del protocolo SSL. La figura 6.19 muestra los resultados obtenidos.

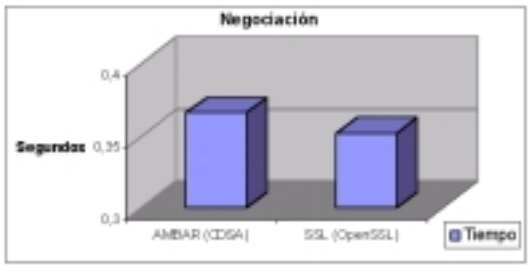

Figura 6.19: Evaluación de la fase de negociación

Como puede apreciarse, el tiempo de ejecución de la negociación de sesiones AMBAR es un 4% superior al empleado por SSL para negociar una sesión de similares características. Hay varias cuestiones que justifican esta ligera diferencia:

- A pesar de que AMBAR tiene una carga criptográfica menor en la fase de negociación que SSL, el tiempo reflejado en la gráfica incluye el intervalo necesario para inicializar los módulos que forman parte de la arquitectura CDSA. Dicho proceso de inicialización implica tanto la carga como la verificación de cada uno de los módulos funcionales de la arquitectura.

- Las llamadas a las funciones de la arquitectura CDSA introducen cierta sobrecarga a la hora de acceder a la funcionalidad ofrecida por alguno de sus módulos. Debe tenerse en cuenta que toda solicitud de servicio pasa por el gestor CSSM, el cual encamina la llamada al gestor del módulo correspondiente y éste a su vez a la implementación concreta del módulo.

A pesar de lo anterior, el resultado obtenido puede considerarse muy destacable ya que es muy similar al rendimiento ofrecido por la implementación de OpenSSL, sobre todo considerando que dicha librería ha estado sometida durante varios años a un proceso continuo de optimización y mejora.

6.6.3 Evaluación de la fase de solicitud y respuesta

Una vez analizada la fase de negociación, el siguiente paso es evaluar el rendimiento relacionado con el intercambio de información de autorización. Para ello, se analizarán dos situaciones distintas que permiten extraer conclusiones acerca de dos aspectos diferentes: por un lado, el rendimiento obtenido en función del número de certificados de credencial transmitidos; por otro lado, el tiempo total asociado a la transmisión de recursos de distinto tamaño. Los datos ofrecidos no incluyen la porción de tiempo necesaria para realizar el cálculo de autorización sino que pretenden ilustrar cuál es la sobrecarga introducida por la fase de intercambio. Más adelante se contrastará este tiempo de transmisión con el tiempo de tramitación de la solicitud.

La figura 6.20 muestra los tiempos de acceso a un recurso, en este caso un certificado SPKI de 1350 bytes, tras el envío de un número variable de credenciales que oscila entre

1 y 16. El intercambio mediante SSL simula el funcionamiento de los módulo RM y ARM del marco AMBAR, puesto que SSL no realiza ningún tipo de distinción respecto al tipo de información que transmite. A partir de la figura puede deducirse que los resultados obtenidos son muy similares (la variación máxima son 10 ms) y que las diferencias existentes están asociadas de nuevo a la sobrecarga introducida por las llamadas CDSA.

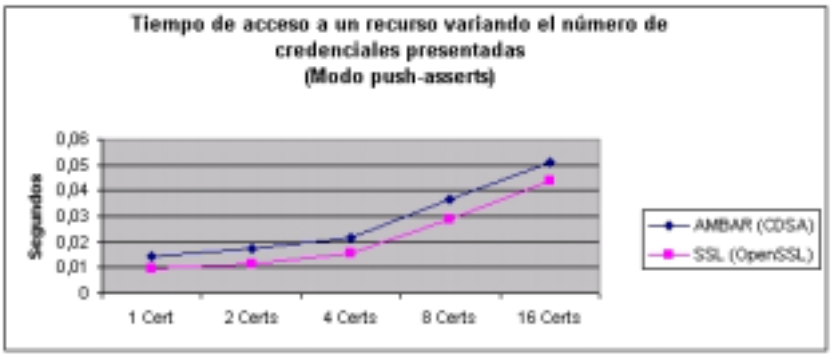

Figura 6.20: Tiempo de acceso en función del número de credenciales

Por otro lado, la figura 6.21 nos permite obtener información acerca de dos cuestiones distintas: en primer lugar, contrasta el rendimiento obtenido por la implementación de AMBAR realizada sobre CDSA frente a una implementación del marco que fue realizada directamente sobre OpenSSL, lo cual permitirá cuantificar la sobrecarga introducida por el *middleware*; en segundo lugar, analiza el tiempo asociado a la transmisión de recursos de mayor tamaño, permitiendo así averiguar si el rendimiento del protocolo se degrada cuando la cantidad de información a transmitir se incrementa.

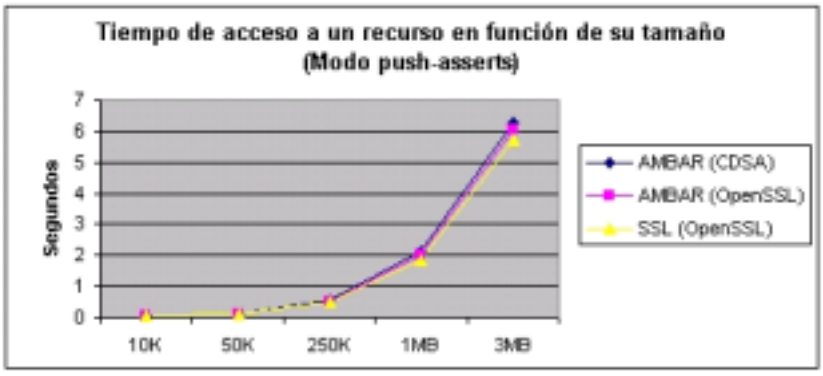

Figura 6.21: Tiempo de acceso en función del tamaño del recurso

En relación con la sobrecarga introducida por CDSA respecto a la implementación de AMBAR basada en OpenSSL, es posible observar que ésta es prácticamente inapreciable ya que apenas supera el 4 % en el peor de los casos. Este hecho demuestra que las ventajas

introducidas por la integración de arquitecturas como CDSA no se ven ensombrecidas por un decremento sustancial en el rendimiento ofrecido por las aplicaciones.

En lo que respecta al comportamiento del protocolo a la hora de transmitir recursos de gran tamaño, la gráfica muestra que el marco AMBAR escala bien, lo que puede ser deducido analizando el incremento lineal del tiempo de transmisión al aumentar también el tamaño del recurso. Al mismo tiempo, puede observarse que los resultados obtenidos son muy similares a los ofrecidos por SSL (la diferencia máxima es de un 6 %).

6.6.4 Evaluación de la tramitación de solicitudes con DCMS

Por último, se analizará el tiempo total de tramitación de una solicitud de certificación realizada mediante las aplicaciones que constituyen el sistema DCMS. Concretamente, las pruebas realizadas implican la participación de la aplicación de las autoridades y de la aplicación de los solicitantes.

El objetivo principal es determinar qué porcentaje de tiempo corresponde al proceso de intercambio de información realizado por el protocolo AMBAR y qué proporción está asociada a la tramitación de la solicitud. A continuación, se detalla cada una de las tareas implicadas en dicho proceso de tramitación, indicando si se engloban dentro de las acciones del protocolo AMBAR o del sistema DCMS:

- Envío de la solicitud y las credenciales relacionadas (AMBAR).
- Verificación de la firma digital de la solicitud (DCMS).
- Verificación de la firma digital de cada una de las credenciales presentadas (DCMS).
- Ejecución del algoritmo que comprueba si la solicitud conforma con la política de la autoridad (DCMS).
- Construcción del certificado de credencial solicitado por el cliente (DCMS).
- Transmisión, por parte de la autoridad, del certificado solicitado (AMBAR).

La figura 6.22 muestra el tiempo total de tramitación obtenido en función del número de credenciales necesarias para satisfacer la política de la autoridad. En ella aparece tanto la porción de tiempo asociada al intercambio AMBAR como al procesamiento DCMS. Como puede apreciarse, el tiempo asociado a los intercambios se incrementa de forma menos severa que el asociado a la tramitación, el cual constituye el 65 % del tiempo total de aquellas solicitudes que necesiten de hasta 16 credenciales distintas para ser satisfechas. Por tanto, el porcentaje ligado al proceso de intercambio va teniendo menor relevancia en cuanto las solicitudes son más complejas, y más peso cuando éstas son más sencillas (cercano al 50 %). Por otro lado, analizando los resultados desde un punto de vista absoluto, es posible comprobar que el tiempo total de tramitación de solicitudes es bastante aceptable al no superar los 150 ms para el caso extremo de 16 credenciales.

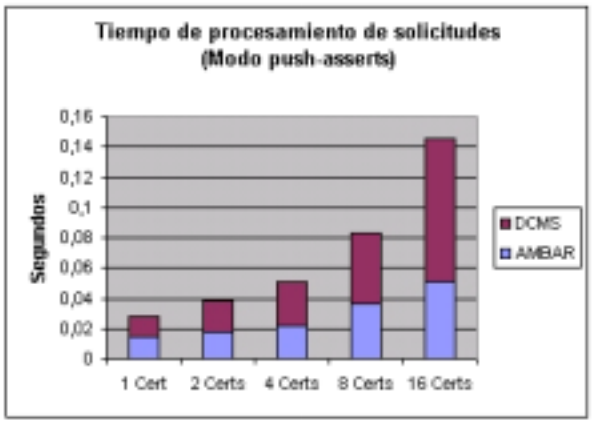

Figura 6.22: Tiempo de tramitación de solicitudes

6.6.5 Conclusiones obtenidas a partir de la evaluación

Las pruebas realizadas permitieron conocer cuáles eran las posibilidades reales de la infraestructura en cuanto a rendimiento se refiere. Como se ha podido comprobar, el protocolo AMBAR ofrece unos resultados muy similares a los aportados por SSL, tanto en su fase de negociación como en la de transmisión de información. Además, se comprobó que la sobrecarga derivada del uso de arquitecturas de seguridad multinivel como CDSA apenas repercutía negativamente sobre el rendimiento total del sistema. Por último, las mediciones relativas al tiempo de tramitación de solicitudes mediante las aplicaciones del sistema DCMS permitieron verificar que el tiempo de respuesta alcanzado se encuentra dentro unos límites que demuestran la aplicabilidad real del sistema, no sólo en cuanto a su diseño sino también en lo que respecta a las prestaciones ofrecidas.

6.7 Conclusiones

Como se puede comprobar a partir de lo descrito en este capítulo, la infraestructura de autorización ha sido desarrollada y verificada con éxito en su papel de middleware de seguridad. En primer lugar, se ha presentado la implementación del marco AMBAR y del sistema distribuido de gestión de credenciales. Tanto la librería AMBAR como el conjunto de aplicaciones DCMS han sido diseñados con el fin de proporcionar servicios de autorización de forma sencilla y estructurada, cuidando que el acceso a su funcionalidad sea lo más completo e intuitivo posible. Para ello, se ha tomado como base la metodología de definición de estructuras de gestión, la cual tiene un fiel reflejo en la forma en la que se articula la implementación de DCMS. Los componentes de la infraestructura se han desarrollado completamente, lo cual implica que toda la funcionalidad descrita en el capítulo anterior (gestión de roles, autorización, reducción, intercambio confidencial de información) se encuentra disponible a la hora de ser aplicada a entornos concretos. Además, se han analizado las prestaciones ofrecidas por dicha implementación, y a la vista de los resulta-

dos obtenidos, es posible afirmar que la infraestructura en su conjunto realiza de forma eficiente las distintas operaciones disponibles, introduciendo una sobrecarga mínima.

En segundo lugar, se ha constatado la viabilidad de estas propuestas a la hora de ser aplicadas a dos escenarios de aplicación completamente distintos. Dicha aplicación de la infraestructura ofrece un conjunto de ventajas adicionales frente al uso de enfoques más tradicionales. Además, la conveniencia de la metodología como directriz a la hora de diseñar sistemas de control de acceso ha sido puesta de manifiesto al modelar con éxito ejemplos concretos de autorización.