J. Lopez, J. A. Montenegro, and J. M. Troya, "Perspectiva Composicional para el Diseño y la Implementación de una PKI", *I Congreso Internacional Sociedad de la Información*, pp. 423-428, 2002.
NICS Lab. Publications: https://www.nics.uma.es/publications

Perspectiva Composicional para el Diseño y la Implementación de una PKI

Javier López, José A. Montenegro y José M. Troya
Dpto.de Lenguajes y Ciencias Computación
E.T.S. Ingeniería Informática, Universidad de Málaga
29071 - Málaga, España
{jlm, crypto, troya}@lcc.uma.es

RESUMEN

El importante papel desempeñado por las Infraestructuras de Clave Pública (PKIs) dentro del marco de la comunicación en Internet, en general, y en el comercio electrónico en particular, nos ha llevado a revisar las actuales pautas seguidas en el desarrollo del software de estos elementos que aportan, tanto seguridad, como confianza al entorno de la certificación digital. En este trabajo presentamos los resultados obtenidos hasta el momento en un proyecto de investigación realizado conjuntamente por el Grupo de Seguridad de la Universidad de Málaga y el Dpto. de Innovación Tecnológica de Banesto, sobre la implementación de una PKI. La particularidad del trabajo es que hemos prestado tanta atención a los aspectos funcionales de la infraestructura como a las técnicas de programación empleadas. Básicamente, hemos enfocado una solución en la que la implementación viene determinada por el creciente auge en el estudio de las arquitecturas software y de los paradigmas surgidos en paralelo como son el caso de la orientación a componentes, los marcos de trabajo y los patrones de diseño. La correcta utilización de estas técnicas nos proporcionan otro punto de vista que nos permite desarrollar todos los elementos de la PKI de forma modular e independiente.

Palabras claves: Seguridad, Firma Electrónica, Certificación Digital, Infraestructuras de Clave pública, Componentes, Patrones de Diseño, Marcos de Trabajo

ABSTRACT

The important role of Public Key Infrastructures (PKIs) inside the general scope of Internet communication, and more precisely, inside electronic commerce, has driven us to the revision of actual procedures followed in the development of software of these elements that provide security and trust to the digital certification environment. In this work we introduce the actual results of a joint research project of the Security Group of the University of Malaga and the Department of Technology Innovation of Banesto regarding a PKI implementation. The originality of this work is that we have paid attention not only to functional aspects of the infrastructure, but also to the programming techniques used. Basically, we have developed a solution in which implementation has been guided by the increase in the study of software architectures and those paradigms that have emerged in parallel, as component orientation, software frameworks, and design patterns. The correct use of these techniques provide a different point of view that allows the development of every PKI building block in a modular and independent way.

Keywords: Security, Electronic Signature, Digital Certification, Public Key Infrastructure, Components, Dessign Patterns, Software Frameworks

1 INTRODUCCIÓN

La tecnología de clave pública está considerada por muchos como criptográficamente más segura y, sobre todo, más fácilmente escalable que la tecnología de clave simétrica. Pero en realidad, la mayor ventaja de los criptosistemas de clave pública es que se pueden utilizar de dos formas: modo de cifrado y modo de autenticación. Este último proporciona la base para la construcción de *sistemas de firma digital* [1], esenciales para la comunicación a través de Internet. Es por ello que esta tecnología está posicionada actualmente en un lugar privilegiado.

Es conocido que el mayor problema de la criptografía de clave pública es el de proporcionar mecanismos que permitan a cualquier usuario *A* obtener la clave pública de otro usuario *B* junto con la prueba de autenticidad de tal clave. La solución típica es la utilización de una *Tercera Parte Confiable*, o más concretamente, de una *Autoridad de Certificación, CA* [2] la cual se encarga de emitir *certificados digitales* para un conjunto determinado de usuarios.

Debido a la existencia de múltiples dominios de usuarios se hace necesaria la coexistencia de muchas autoridades, estableciéndose relaciones de confianza entre ellas. Tal multiplicidad ocasiona que el esquema normal de certificación que enlaza a un usuario con su correspondiente autoridad se utilice de forma recursiva para poder enlazar usuarios de dominios distintos. Con ello se crean *caminos de certificación*, partiéndose de la clave de una Autoridad en la que el usuario deposita una confianza directa y, a través de certificados en los que se tiene una confianza inducida, llegar hasta la clave del usuario destino.

El diseño de una infraestructura de CAs que permita a los usuarios el establecimiento de esas cadenas de confianza se denomina *Infraestructura de Clave Pública, o PKI* [3,4]. Este es el marco subyacente que permite que la tecnología de clave pública se pueda realmente implantar de un modo extenso pues proporciona la base confiable necesaria para la correspondencia electrónica entre aquellos usuarios que no pueden intercambiar manualmente sus claves.

Así, a través de una PKI, y mediante la administración de los certificados de claves públicas, se puede establecer y mantener un entorno de red seguro, posibilitando el uso de servicios como los de cifrado y de firma digital en una amplia gama de aplicaciones.

El diseño y la implementación de distintos modelos de PKIs, así como la implantación de estas infraestructuras dentro de una extensa gama de aplicaciones para Internet, han sido áreas de investigación y desarrollo muy prolíficas en los últimos años dentro del mundo de la Seguridad de la Información. Esto ha provocado que en poco tiempo el número de PKIs desarrolladas por compañías privadas, así como por centros universitarios y de investigación, se haya disparado, derivando en una gran oferta de productos comerciales basados en PKI.

La mayoría de esas PKIs, y especialmente las desarrolladas para el ámbito comercial, se basan en una serie de estándares, protocolos y normas aprobadas o reconocidas internacionalmente. De esto se podría deducir, en teoría, que sus núcleos han de ser funcionalmente idénticos, y que las mayores diferencias entre ellas aparecen a nivel de la interfaz de usuario (entendiendo por tal *usuario final*, *Operador de CA*, *Administrador de CA*, etc.), y que, por lo tanto, es precisamente ahí donde las casas comerciales intentan introducir un mayor elemento diferenciador para conseguir posiciones avanzadas de mercado.

Como se ha mencionado, esa es la teoría, porque la práctica muestra que no hay dos PKIs iguales, con independencia de las diferencias que ya se presuponen respecto al interfaz de usuario. No cabe duda de que a pesar de seguir los mismos estándares, distintas PKIs funcionan internamente de forma diferente, incluso para las funciones más básicas.

Las diferencias en las técnicas de programación seguidas por los distintos grupos de programadores se perciben, en ocasiones, diametralmente opuestas. Esta situación viene influenciada, sin duda, por la mayor profundización de los equipos de desarrollo en todos aquellos elementos relativos a la seguridad del sistema, y una prácticamente nula preocupación en los aspectos básicos de diseño que pueden proporcionar una correcta implementación. Esta situación posibilita incluso la existencia de fallos en el software que, con el tiempo, se convierten precisamente en puntos de ataque al sistema.

En este artículo presentamos los resultados obtenidos hasta el momento en un proyecto de investigación realizado conjuntamente por el Grupo de Seguridad de la Universidad de Málaga y el Dpto. de Innovación Tecnológica de Banesto, sobre la implementación de una PKI, con la particularidad de haber prestado tanta atención a los aspectos funcionales de la misma como a las técnicas de programación empleadas, con el objeto de evitar los problemas referidos anteriormente.

El enfoque viene influenciado por el creciente auge en el estudio de la arquitectura software y los paradigmas surgidos paralelamente como son el caso de la orientación a componentes, los marcos de trabajo [5] y los patrones de diseño [6]. La correcta utilización de estas técnicas nos proporcionan otro punto de vista que nos permite desarrollar todos los elementos del sistema de forma modular e independiente.

Más concretamente, se han aplicado técnicas de programación orientada a componentes, las cuales han permitido construir una PKI en las que cada una de sus partes (o componentes) puede ser intercambiada por otras ya desarrolladas con anterioridad, o por desarrollar en un futuro, sin que esos cambios parciales afecten al funcionamiento del resto de la infraestructura.

A modo de ejemplo, sirva decir que la PKI puede funcionar con distintas librerías criptográficas, e incluso con distintas librerías gráficas, sin prestar excesiva atención a quienes han sido los desarrolladores de tales librerías. Esta modularidad permite al diseñador de la PKI la sustitución de unos componentes por otros a medida que nuevas y más depuradas versiones de librerías van apareciendo en el mercado, de tal forma que la PKI nunca queda obsoleta. De la misma forma, se pueden añadir fácilmente librerías de desarrollo propio para cualquiera de esos componentes si el diseñador estima que alguna de las funcionalidades de la PKI no está correctamente orientada en el software comercial del que dispone en un momento determinado.

El artículo se estructura de la siguiente forma. En la sección 2, realizamos una descomposición funcional de los múltiples elementos que componen una PKI, describiendo someramente sus principales cometidos. Las distintas causas que han motivado un desarrollo de PKI basado en componentes se explican en la sección 3. Posteriormente, en la sección 4, se describen con detalle cada uno de los componentes que integran la arquitectura, esbozando el contenido y la finalidad de cada uno de ellos en los correspondientes subapartados. Finalmente, en la sección 5 se exponen las conclusiones, así como la futura investigación que tiene como punto de inicio el presente trabajo.

2 ESTUDIO FUNCIONAL DE LOS ELEMENTOS DE UNA PKI

La implementación de una PKI desde la perspectiva de su funcionalidad viene prácticamente determinada a partir de la información aportada por los estándares, RFCs (Request for Comments) de la IETF (Internet Engineering Task Force) [7], normas PKCS (Public Key Cryptographic Standards) [8], etc. La figura 1 muestra con bastante detalle los distintos elementos que forman un núcleo de PKI, así como la interacción entre ellos y las principales funciones involucradas.

Se puede observar que cualquier desarrollo basado en los aspectos de funcionalidad distingue las siguientes partes generales:

- *Autoridad de Certificación* (CA): Este segmento es el encargado de realizar todas las tareas relacionadas con la gestión de certificados digitales de identidad y la emisión de listas de revocación. El segmento, que se corresponde en la figura con el núcleo central denominado "CA" incluye desde la función de generación de claves y certificados hasta las tareas de almacenamiento de los últimos.
- *Flujos de comunicación externa*: Los elementos del sistema externos a la CA han de estar correctamente comunicados con la misma para permitir el adecuado funcionamiento de la infraestructura como un todo. Este segmento abarca todos los flujos (representados por flechas) que salen del núcleo hacia las entidades como la *Autoridad de Registro* (RA), los usuarios finales, el operador de CA, e incluso el directorio X.500.
- *Interfaces gráficos de los elementos del sistema*: Este elemento tiene como misión ensamblar muchas de las funcionalidades desarrolladas en las dos partes anteriores, proporcionando operatividad completa a la PKI mediante la introducción de elementos gráficos que facilitan la labor tanto a las entidades mencionadas, como a los clientes que hagan uso del sistema.

Si prestamos atención al conjunto de funciones realizadas en las distintas partes en que se divide la infraestructura, encontramos que se comparte la ejecución de rutinas bastantes similares. La implementación de tales rutinas mediante el uso de técnicas de orientación a objetos ahorraría en gran medida el tiempo de desarrollo de rutinas con un cometido similar [9].

Más aún, si lo que pretendemos es obtener una solución cuyas partes sean reutilizables y fácilmente actualizables, es necesario abstraer la funcionalidad de las mismas, de tal forma que se obtenga un sistema totalmente operativo sin estar sujeto a la implementación de las rutinas básicas de soporte.

Esta idea general es la base del presente trabajo, y se concreta con mayor detalle en las distintas subsecciones del apartado siguiente.

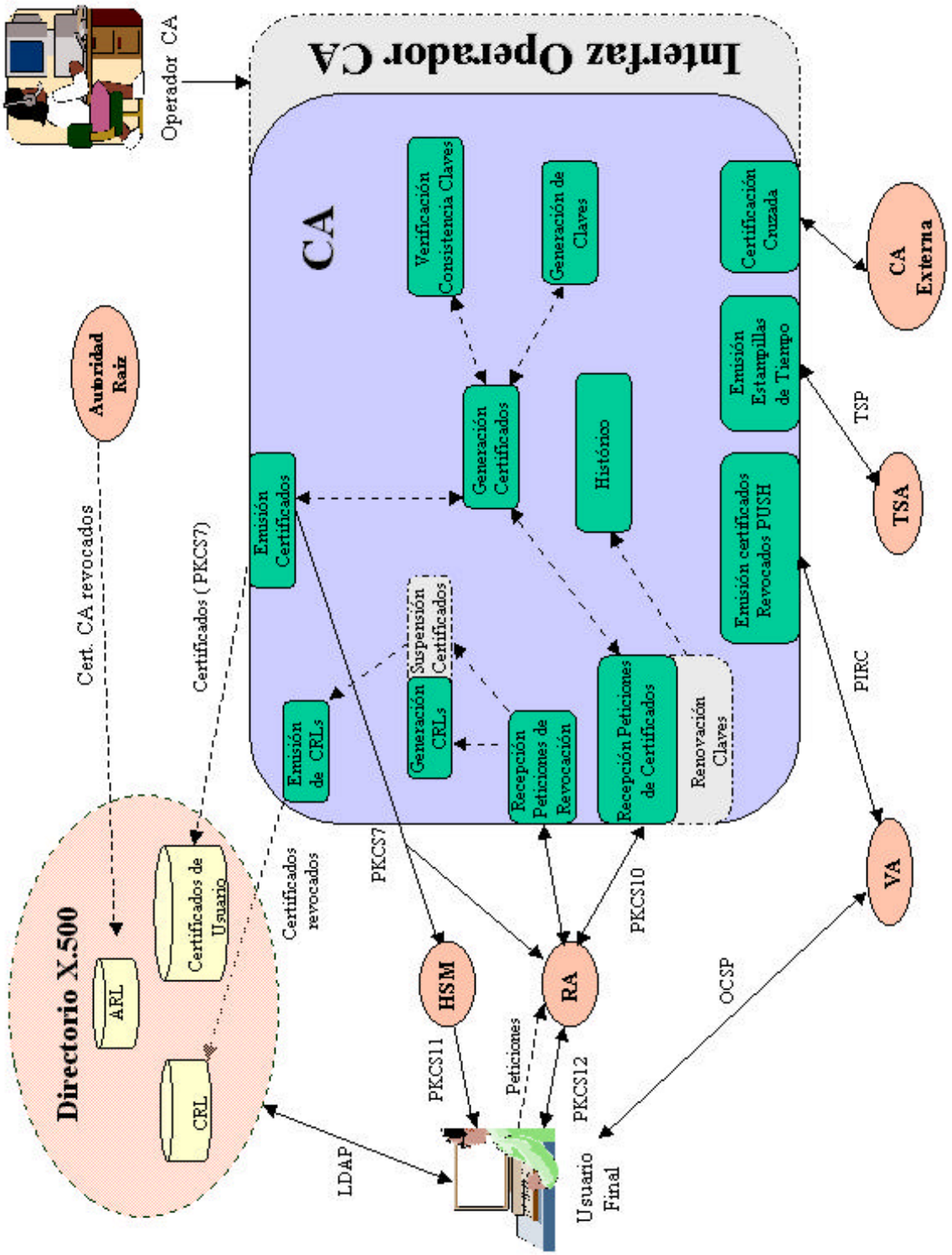

Figura 1. Detalle de las partes y flujos de comunicación de un núcleo de PKI

3 ENFOQUE COMPOSICIONAL DE LOS ELEMENTOS DE UNA PKI

Las necesidades software para la creación de una PKI son bastante elevadas, ya que unido al hecho de tener que usar las rutinas criptográficas que se requieren, existen unos protocolos, normativas y reglas de codificación [10] que se deben seguir si se pretende que el resultado esté ajustado a los estándares. Debido a la reciente y explosiva implantación de este campo de la Seguridad de la Información, se sufre de una constante evolución en los estándares a utilizar, por lo que se presume difícil la implementación de un sistema de la complejidad que nos ocupa partiendo desde cero. De lo contrario, se corre el riesgo de que el desarrollo quede obsoleto antes de llegar a una primera fase de implantación. Más aún, con una alta probabilidad, algunos de los elementos implementados pueden caer en desuso antes de esa fase.

Estos argumentos llevan a la necesidad de apoyar el desarrollo de software de la PKI con software previamente desarrollado por terceros, especialmente si tal software ha sido previamente usado y mantenido por una gran comunidad de usuarios.

Uniendo todas las piezas de las que se ha tratado en las páginas anteriores, se presenta la necesidad y justificación de la aplicación de la Ingeniería del Software, así como de sus elementos ya mencionados, al diseño y desarrollo de aplicaciones de seguridad, para convertirlas en un software fiable y de calidad [11].

Partiendo de nuevo del esquema de la figura 1, y aplicando un enfoque composicional, hemos distinguido cuatro componentes dentro de un marco de trabajo de seguridad mucho más complejo y funcional actualmente en desarrollo dentro del Proyecto de Investigación citado en la introducción del artículo. Esos componentes son los relativos a: (a) el núcleo criptográfico, (b) el almacenamiento de información, (c) los protocolos de comunicación, y (d) el interfaz gráfico.

Por lo tanto, la implementación realizada se ha basado en esos cuatro módulos, los cuales han sido creados siguiendo una misma filosofía, la de abstraer lo más posible su interfaz de su correspondiente implementación. Esto se ha llevado a cabo mediante la utilización de patrones de diseño; más concretamente, diseñando clases abstractas que proporcionen una interfaz común y, además, la creación de tantas clases hijas de la clase abstracta como diferentes librerías y APIs se quieran utilizar.

Esta solución, costosa en su diseño, permite establecer los pilares sobre los que recaerá toda la arquitectura, tolerando el uso de las distintas librerías disponibles, o bien, si acontece, realizar una implementación propia de las rutinas.

El siguiente apartado muestra los detalles de los componentes referidos.

4 DETALLES DE LOS COMPONENTES

4.1 Núcleo criptográfico

Su tarea principal es la de cubrir las principales tareas criptográficas del sistema como son la creación, emisión y administración de elementos criptográficos tales como pares de claves <pública-privada>, certificados X509 v3 [12], Solicitud de Petición de Certificados o CSRs [13], administración de listas de revocación, etc.

La realización de este componente ha conllevado el estudio y prueba de varias librerías criptográficas de libre distribución, concretamente, *RSAEuro*, *Cryptlib* y *OpenSSL*. Un estudio comparativo de las características de cada una de ellas ha determinado que, entre estas, la más adecuada resultaba ser la librería *OpenSSL* [14]. Además, tal estudio comparativo ha proporcionado el conocimiento acerca de la división lógica del interfaz de programación de las librerías, que es el suministrado a los usuarios, y el núcleo encargado de proporcionar la funcionalidad requerida.

La figura 2 muestra cómo hemos utilizado esta división para desarrollar los componentes. Se indica como, en el caso relativo al núcleo criptográfico, sólo se ha necesitado implementar una capa muy pequeña, denominada *Interfaz de Adaptación*, para que cualquier librería quede integrada correctamente en el componente.

Si nos fijamos con detalle en lo que esto supone, estaríamos abocados a los innumerables problemas de compatibilidad que normalmente surgen en la generación de elementos criptográficos, pues se permitirían cosas tan dispares como generar claves con una librería y, a continuación, hacer uso de éstas para la emisión de certificados con otra librería diferente. Sin embargo, con la solución que proponemos es posible elegir los mejores elementos de cada librería, sin tener que desvincularlos de su entorno natural.

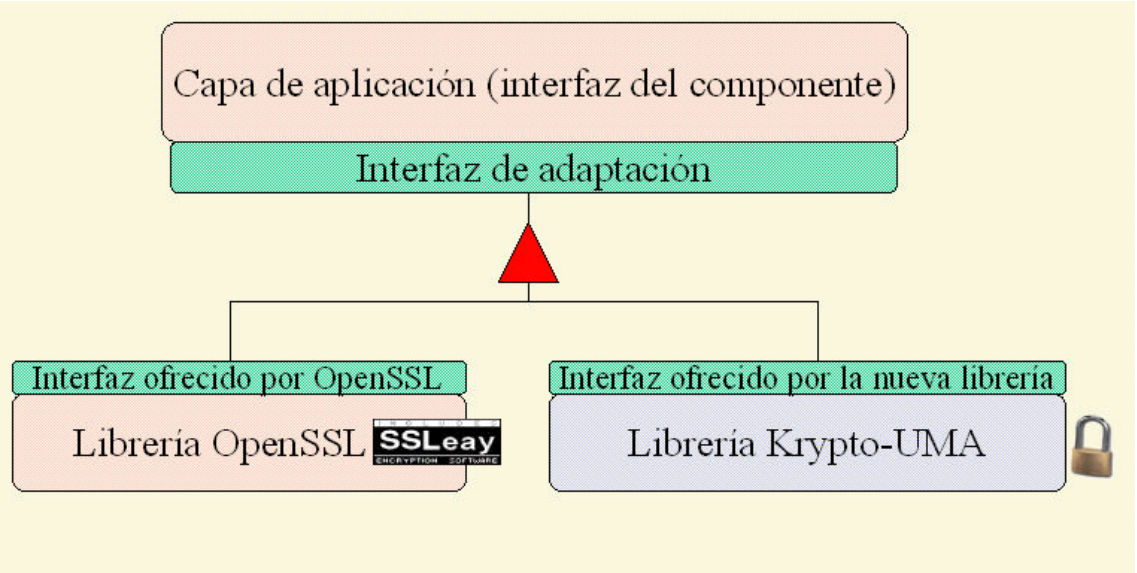

Figura 2. División de componentes en el interfaz de programación del núcleo criptográfico

4.2 Almacenamiento de información

Dentro del amplio concepto de almacén intentamos reunir aquí todos los posibles medios de almacenamiento necesarios, por las entidades pertenecientes al sistema, desde la Autoridad de Certificación hasta los clientes.

El enfoque seguido es el de intentar abarcar el mayor número de medios de almacenamientos en el desarrollo, para facilitar todo lo relativo a la acción de guardar cualquier objeto (estructura de información), abstrayéndolo del medio físico o dispositivo, que puede ser seleccionado por el usuario con posterioridad.

Otro elemento muy importante a tener en cuenta, y del cual también se ocupa el componente de almacenamiento de información, es el de la codificación de los objetos almacenados, que como bien es sabido pueden ser introducidos y recuperados en distintos formatos, entre los que destacan BER y DER [15].

Los tipos de dispositivos de almacenamiento contemplados en el trabajo van desde los más comunes, como el almacenamiento magnético en disco duro o disquetes, hasta las tarjetas inteligentes, pasando por un servicio de directorio X.500 distribuido. En resumen, se puede incluir cualquier dispositivo que se ajuste al interfaz especificado por el componente.

La figura 3 muestra la especificación formal en UML [16] de la jerarquía de clases necesarias para realizar la implementación del componente de almacenamiento de información, pudiéndose observar, además, cómo realizamos una sobrecarga de los métodos *extraer* y *almacenar*. Esto permite llevar a cabo las operaciones de almacenamiento y recuperación de distintos tipos de información, sin tener que recordar un gran número de métodos para realizar las distintas acciones ni los métodos necesarios para establecer y consultar la codificación empleada.

Se observa claramente, en la figura 3, que el tratamiento es idéntico tanto para el almacenamiento local como para el almacenamiento remoto (caso de los directorios X.500), con lo que se elimina la necesidad de conocer al detalle los protocolos necesarios para la comunicación, así como tampoco otros elementos que incomodan la tarea del administrador y de los usuarios.

4.3 Protocolos de Comunicación

Actualmente el campo de las comunicaciones en Internet es el que está teniendo un mayor auge dentro de las aplicaciones informáticas, y ha conseguido poner en funcionamiento muchas soluciones para sistemas distribuidos que se habían dejado de usar. El causante de este cambio ha sido la popularidad adquirida por los protocolos de la pila de TCP/IP [17], desde el sobradamente conocido *http*[18] hasta el sencillo y no menos útil *pop3* [19].

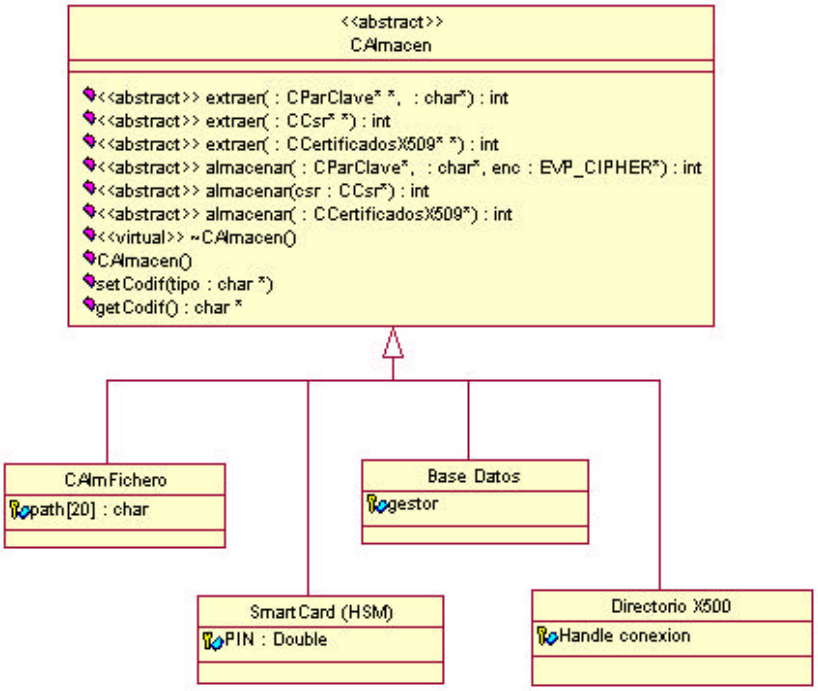

Figura 3. Especificación formal de la jerarquía de clases del componente de almacenamiento de información

La correcta programación de los *sockets* [20] es tediosa, y en ciertos casos requiere de los programadores unos conocimientos más amplios sobre las comunicaciones en TCP/IP de lo que normalmente se suele tener. Actualmente, existe la tendencia de aplicar los mecanismos de orientación a objetos a las comunicaciones, y más aún, se están estableciendo patrones de diseño que nos evitan realizar tareas repetitivas. Como consecuencia nos liberan de potenciales errores de programación.

La idea básica es descargar al programador de todas las tareas relacionadas con la comunicación, aportándole una rígida plataforma, exenta de errores, de tal forma que se pueda centrar únicamente en el estudio y posterior definición del protocolo, pues es la plataforma la que le detalla las propiedades de cada protocolo a implementar.

A la hora de llevar a cabo la implementación de la PKI, ha sido necesario usar un número no desdeñable de protocolos para realizar las tareas relativas a la gestión de la infraestructura, así como la comunicación entre sus elementos. La realización de dichos protocolos ha puesto de manifiesto la necesidad de incluir métodos de codificación, tal y como ocurriera en el componente de almacenamiento de información, puesto que ciertos protocolos se basan en la emisión y recepción continuadas de mensajes codificados en DER, cómo es el caso de OCSP [21].

Debido a la controversia existente actualmente sobre la verificación de la certificados, se están desarrollando nuevos protocolos, como SCVP [22], y revisando otros, como OCSP del cual se está concluyendo su versión 2 [23]. En ambos casos se requiere de una rápida implantación debido a los delicados procesos en los cuales se utilizan. Precisamente, la perspectiva composicional defendida en este trabajo permite aplicar la celeridad necesaria para que los protocolos que van apareciendo no se queden en meras especificaciones formales, sino que, muy al contrario, cumplan estrictamente su cometido.

4.4 Interfaz Gráfico

Los interfaces gráficos en esta última década han permitido acercar, de manera eficiente, toda clase de tecnología de la información hacia usuarios finales que carecían de conocimientos técnicos adecuados. Inicialmente cada programador debía crear su propio interfaz gráfico para cada aplicación que desarrollaba, lo que además de hacer los proyectos inviables, no facilitaba su uso al cliente, debido principalmente a la enorme disparidad existente entre distintos productos.

La revolución surgida mediante la inclusión del sistema operativo *Windows* en el mercado y la aportación de esta plataforma de las herramientas necesarias para el desarrollo de aplicaciones visuales en periodos muy cortos de tiempo, ha cambiado totalmente la perspectiva de desarrollo de las aplicaciones. Esa revolución ha dado paso a un periodo en el que sistemas operativos de libre distribución, como *Linux*, han hecho uso de una correcta metodología de diseño, lo que le han permitido entrar en competencia directa con aquello que comercialmente estaba más implantado.

Por lo tanto, hemos tenido muy en consideración el diseño visual, apostando en nuestro trabajo por una librería de libre distribución totalmente portable entre plataformas como es el caso de GTK [24]. A pesar de ello, también aquí hemos seguido la filosofía de la orientación a componentes, no vinculándonos totalmente con la solución implementada

Esta filosofía es simbolizada en la figura 4, donde podemos observar la necesidad de creación de una jerarquía de clases por cada elemento perteneciente al interfaz gráfico, mostrando en este caso el conjunto de clases referente a los botones y siendo el punto de partida del patrón factoría creado para completar la funcionalidad del componente.

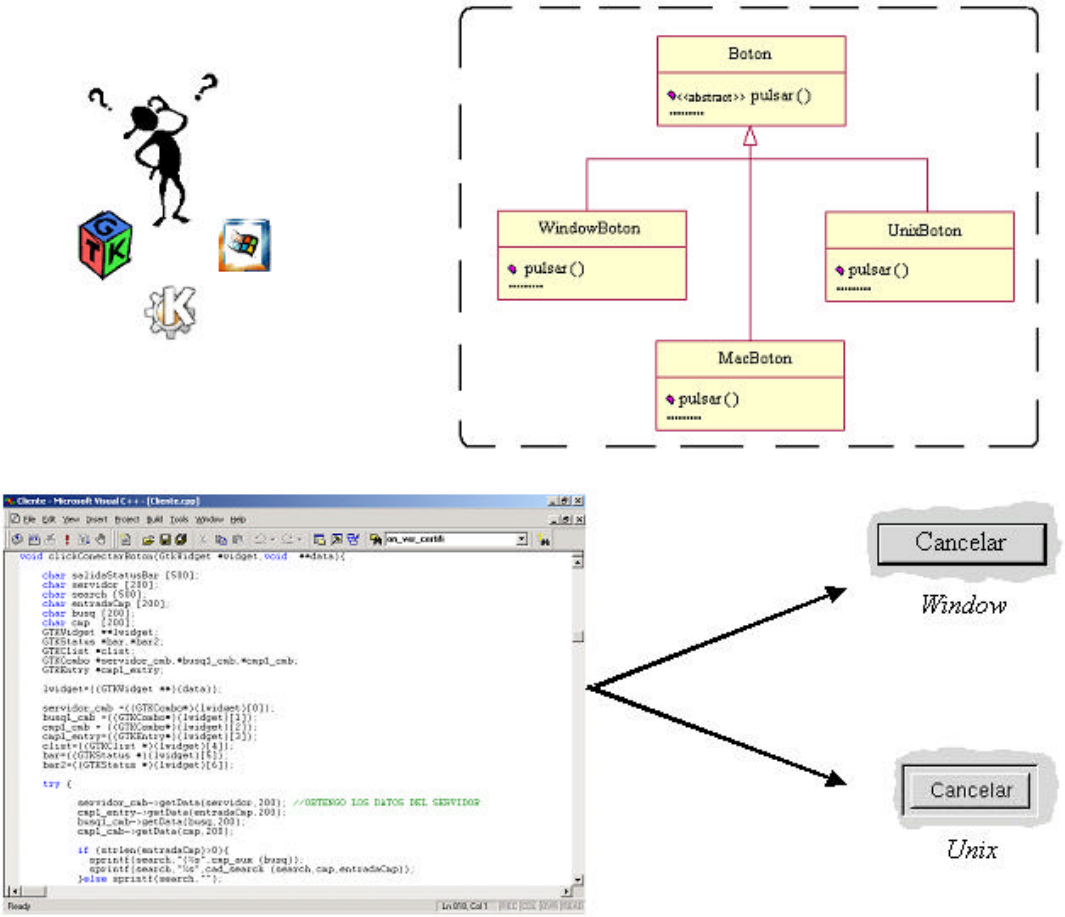

Figura 4. Jerarquía de clases de los elementos del interfaz gráfico

5 CONCLUSIONES

Actualmente, las necesidades software para la creación de una PKI son bastantes elevadas, ya que debemos cumplir unos inauditos requisitos de calidad y fiabilidad. Todo esto unido a la reciente y explosiva implantación de este campo de la Seguridad de la Información, hace que se requiera un desarrollo veloz y fiable.

La correcta utilización de las técnicas introducidas por la Ingeniería del Software, como son el caso de la programación orientada a objetos, componentes y marcos de trabajo, proporciona un buen punto de partida para la realización de un diseño e implementación modular e independiente, que cumpla los requisitos de desarrollo anteriormente mencionados.

Este trabajo cubre el diseño e implementación de un núcleo de PKI orientado a componentes, siendo la primera piedra para la creación de un marco de trabajo más amplio, el cual cubrirá las necesidades software relativas a la Seguridad de la Información.

La solución actual está compuesta por cuatro módulos, *Núcleo criptográfico*, *Almacenamiento de información*, *Protocolos de Comunicación* e *Interfaz Gráfico*, detallados a lo largo del presente artículo, que proporcionan la funcionalidad requerida por el núcleo de PKI, además de ser posibles elementos constituyentes en proyectos futuros, gracias a su diseño basado en la reutilización.

6 REFERENCIAS

1. Pfitzmann, Birgit; *Digital Signature Schemes*, Springer-Verlag 1996
2. Feghhi, Jalal; Williams, Peter, *Digital Certificates*, Addison-Wesley, 1999
3. Ford, W. *Secure Electronic Commerce*, Prentice Hall,1999
4. Grupo de trabajo IETF PKIX. http://www.ietf.org/html.charters/pkix-charter.html
5. Larsen, Grant. *Designing Component-Based Frameworks Using patterns in the UML*. Communications of the ACM, Octubre 1999.
6. Gamma, Erich; et al. *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley,1994
7. http://www.ietf.org/
8. http://www.rsasecurity.com/
9. Meyer, Bertrand; *Construcción de Software. Orientado a Objetos*. Prentice Hall, 1998
10. Kaliski, Burton. *A Layman's Guide to a Subset of ASN.1, BER, and DER*. Noviembre 1993
11. Pressman, R. S. *Ingeniería del Sofware. Un Enfoque Práctico*. 4ª Ed. McGraw Hill. 1998.
12. M. Myers,C. Adams,D. Solo, D. Kemp.*Internet X.509 Certificate Request Message Format*, Internet Request Comments: 2511, Marzo 1999

13.R. Housley, W. Ford, W. Polk D. Solo. *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*, Internet Request Comments: 2459, Enero 1999

14. http://www.openssl.org
15. *CCITT Recommendation X.208: Specification of Abstract Syntax Notation One (ASN.1)*, 1988.
16. Booch, Grady; Rumbauch, James; Jacobson, Ivar. *El lenguaje unificado de modelado* (UML), Addison-W, 1998
17. Comer, D; Stevens, D. *Internetworking with TCP/IP Vol 1,2,3*. Prentice Hall
18. R. Fielding, et al. *Hypertext Transfer Protocol – HTTP/1.1. Internet Request Comments: 2616*, Junio 1999
19. R. Nelson.*Some Observations on Implementations of the Post Office Protocol*. Internet Request Comments:1957 (POP3). Junio 1996
20. Stevens, W. Richard. Unix, *Network Programming, Volumen 1,2*. Prentice Hall, 1998

21. Schmidt, Douglas; et al. *Patterns for Concurrent and Networked Objects*, Wiley, September 2000

22. Myers, M.; et al. *Online Certificate Status Protocol. OCSP.* Internet Request Comments: 2560, Junio 1999

23. Malpani, Ambarish; et al. *Simple Certificate Validation Protocol (SCVP)*. Internet Draft, Julio 2001

24. Ankney, Rich; et al. *Online Certificate Status Protocol, version 2*. Internet draft, Marzo 2001

25. http://www.gtk.org