J. A. Montenegro, J. Lopez, and F. Moya, "Diseño e Implementación del Marco de Trabajo de Certificado de Atributos X509 para la Delegación de Privilegios", *VIII Reunión Española sobre Criptología y Seguridad de la información (VIII RECSI),Madrid*, pp. 571-581, 2004.
NICS Lab. Publications: `https://www.nics.uma.es/publications`

# Diseño e Implementación del Marco de Trabajo de Certificados de Atributos X509 como plataforma para la Delegación de Privilegios*

Javier López, José A. Montenegro, Fernando Moya

Departamento de Lenguajes y Ciencias de la Computación,
E.T.S. Ingeniería Informática, Universidad de Málaga,
{jlm,monte,fmoya}@lcc.uma.es

**Resumen** Este trabajo muestra los detalles de una implementación prototipo del marco de trabajo de Certificados de Atributos X.509 (Xac), propuesto por la recomendación ITU-T. La implementación utiliza como base de la plataforma la librería OpenSSL, que ha sido escogida por las ventajas que presenta en relación con otras librerías. El artículo detalla los componentes de la implementación del marco de trabajo, centrándose en el modelo de delegación especificado por la propuesta ITU-T, y tomando en consideración el reciente informe emitido por ETSI sobre Xac.

## 1. Introducción

Los métodos de control de acceso estándar normalmente obligan a que el usuario posea los privilegios en un sistema para acceder a los recursos de ese dominio. Este tipo de sistemas utilizan los mecanismos de control de acceso que le brinda el sistema operativo, y necesitan que todos los usuarios de un recurso compartido estén localmente registrados en el sistema. Esta solución, utilizada normalmente en los entornos actuales por su simplicidad, no es una solución válida en los sistemas distribuidos de hoy en día, debido a su falta de escalabilidad.

Actualmente, existen un número elevado de sistemas donde la autorización es controlada por elementos que son externos a los elementos funcionales del propio sistema. En estas situaciones, es necesario establecer una relación de confianza entre los elementos funcionales y los elementos responsables de la autorización.

Este planteamiento permite que varios sistemas compartan los mismos elementos encargados de la autorización, dando lugar a un sistema más escalable que aquellos que utilizan los tradicionales métodos de control de acceso. Este tipo de configuración es denominada *autorización distribuida*: los sistemas de autorización publican los elementos de autorización y los sistemas funcionales obtienen estos elementos para evaluar las acciones de los usuarios. Inicialmente,

* Este trabajo ha sido parcialmente financiado por el Ministerio de Ciencia y Tecnología bajo el Proyecto TIC2003-08184-C02-01

los atributos o privilegios pertenecen a un sistema funcional. Estos atributos son delegados a un sistema de autorización, que es el encargado de administrarlos. Por todo ello, se deduce que cada sistema funcional debe confiar en un sistema de autorización que sea capaz de utilizar los atributos delegados.

Actualmente existen varias propuestas que intentan resolver la situación anterior, y aunque no resuelven completamente los problemas de delegación, constituyen un buen punto de partida para diseñar un sistema de delegación. Las propuestas a las que hacemos mención son AAAARCH [10], Akenti [12], Kerberos [18], Sesame [9], y SDSI/SPKI [4]. Como posteriormente se hará explicito, nuestra implementación del marco de trabajo de autorización está basada en la recomendación de la ITU-T X.509, centrándose en las funciones de delegación.

El artículo está estructurado de la siguiente forma, la sección 2 presenta las ventajas que posee el Certificado de Atributo X.509, ante otras soluciones, como elemento para transportar la información de autorización, preservando los requisitos de seguridad necesarios. La sección 3 describe las acciones necesarias para añadir el certificado de atributo X.509 a la librería OpenSSL y las ventajas que aporta esta librería con respecto a otras librerías criptográficas existentes en el mercado. En la sección 4, presentamos los componentes del sistema que son necesarios desarrollar para realizar el marco de trabajo. Finalmente, la sección 5 finaliza el trabajo exponiendo las conclusiones y las líneas de trabajo futuras.

## 2. Estructuras para almacenar la información de Autorización

Los sistemas de autorización necesitan hacer uso de información de autorización para realizar sus pertinentes tareas. Esta información debe ser especificada en un formato que garantice ciertas propiedades como autenticación, confidencialidad, integridad y no repudio. Las soluciones actuales han adoptado diversos formatos para transportar la información de autorización como es el caso de los tokens, web cookies o certificado de identidad. Estos formatos presentan varios inconvenientes:

- *Soluciones Propietarias.* Las soluciones comerciales usualmente desarrollan su propia estructura, como es el caso de los tokens o las cookies, para almacenar la información de autorización. Normalmente, los formatos propietarios presentan numerosas debilidades que pueden producir vulnerabilidades en el sistema completo.
- *Certificados de identidad.* Esta solución da lugar que las entidades de atributos sean dependientes de las entidades de identidad, situación poco deseable. Primero, es un inconveniente en las situaciones donde la autoridad que emite el certificado de identidad no es la autoridad encargada de la asignación de los privilegios. Segundo, incluso en las situaciones donde la autoridad es la misma, debemos considerar que el intervalo de validez de las identidades es relativamente más duradero cuando lo comparamos con la frecuencia que

se modifican los privilegios del usuario. Por tanto, cada vez que se modifiquen los privilegios es necesario revocar el certificado de identidad, y es sobradamente conocido que el proceso de revocación es altamente costoso. Además, muchas aplicaciones tratan con mecanismos de autorización como es el caso de la delegación (traspaso de privilegios desde una identidad que posee unos privilegios a otra entidad) o de la sustitución (un usuario es temporalmente substituido por otro usuario, y este usuario obtiene los privilegios del primero durante un cierto periodo de tiempo). Los certificados de identidad no soportan estas características.

Por otro lado, la propuesta de la ITU-T es la primera en utilizar el concepto de *Certificado de Atributos X.509(Xac)* en 1997 [6]. Posteriormente, una segunda versión de *Xac* fue propuesta para mejorar la anterior versión[7].

Una vez desarrollado por parte de ITU-T, IETF ha realizado un RFC [5] donde incluye recomendaciones acerca de como usar los campos de la estructura de *Xac* en escenarios que hagan uso de Internet.

El certificado es utilizado para transportar la información referente a los privilegios de las entidades involucradas en el sistema, siendo además el elemento básico para desarrollar la *Infraestructura de Administración de Privilegios* (PMI).

## 3. Utilizando la librería OpenSSL para implementar Certificados de Atributos X.509

OpenSSL es una librería criptográfica basada en el proyecto SSLeay. El proyecto SSLeay fue inicializado por Eric A. Young y Tim J.Houston [19], y como propuesta inicial pretendía ofrecer el protocolo SSL para cualquier aplicación de seguridad. El proyecto OpenSSL fue iniciado en 1995, y actualmente es la librería criptográfica mayoritariamente usada. Las principales razones son:

- Es distribuida usando una licencia de código libre. Esta situación permite a los usuarios realizar los cambios y optimizaciones en los elementos criptográficos que la componen.
- Es posible integrarla en cualquier aplicación comercial sin necesidad de abonar tasas, permitiendo ofrecer soporte SSL criptográfico, comparable con cualquier librería comercial.
- La librería puede ser ejecutada en numerosas plataformas. Esta característica permite exportar las aplicaciones que utilizan la librería a cualquier sistema operativo.

La importancia de esta librería es avalada por su inclusión en varios proyectos de relevancia. La librería es el núcleo de Apache Seguro [15], el servidor web seguro más ampliamente utilizado. Adicionalmente, es usado por el navegador Opera [17] y en proyectos como OpenSSH [16], OpenCA [3], y XMLSec [11],etc.

Estas razones nos han influenciado a escoger la librería OpenSSL como plataforma para desarrollar nuestro sistema. Sin embargo, la actual versión de

OpenSSL no proporciona soporte para *Xac*. Por tanto, la primera tarea a realizar es incluir el soporte de *Xac* en la librería.

### 3.1. Como Añadir los Certificados de Atributos X.509 en OpenSSL

OpenSSL tiene dos partes principales: una librería criptográfica (*crypto*) y una librería SSL(*ssl*). Estas elementos son diferenciables en el código fuente y en las librerías dinámicas resultantes después de completar los pasos aquí indicados.

Incluir *Xac* en OpenSSL requiere únicamente de la modificación de la librería criptográfica, debido a que *crypto* define los algoritmos y estándares criptográficos. Cada elemento criptográfico es localizado en un directorio que contiene las fuentes necesarias para su compilación.

Por tanto, el proceso general de incluir *Xac* en OpenSSL ha sido divido en tres fases:

**Primera Fase: Definición de la estructura de los certificados** Esta fase implementa la estructura *Xac*, traduciendo la definición ASN.1[8] a código fuente en el lenguaje C. La definición completa de la definición en ASN.1 puede ser encontrada en [7] y [5].

OpenSSL utiliza un mecanismo de macros para traducir el esquema ASN.1 a código C. La siguiente estructura es un ejemplo que muestra una posible definición de emisor:

```
IssuerSerial  ::=  SEQUENCE {
          issuer         GeneralNames,
          serial         CertificateSerialNumber,
          issuerUID      UniqueIdentifier OPTIONAL
}
```

El resultado de aplicar el sistema de macros a la estructura anterior es el siguiente código en el lenguaje C:

```
ASN1_SEQUENCE(X509AT_ISSUER_SERIAL) = {
        ASN1_SIMPLE(X509AT_ISSUER_SERIAL, issuer, GENERAL_NAMES),
        ASN1_SIMPLE(X509AT_ISSUER_SERIAL, serial, ASN1_INTEGER),
        ASN1_OPT(X509AT_ISSUER_SERIAL, issuerUID, ASN1_BIT_STRING)
        } ASN1_SEQUENCE_END(X509AT_ISSUER_SERIAL)

IMPLEMENT_ASN1_FUNCTIONS(X509AT_ISSUER_SERIAL)
```

**Segunda Fase: Definición de las funciones asociadas** En la fase anterior, hemos obtenido una estructura en código C, sin embargo, es necesario definir un conjunto de funciones para acceder desde la aplicación del usuario a la información que almacena el *Xac*

El mecanismo de macros ofrece otra ventaja, internamente define las funciones usadas para codificar la estructura del certificado a codificación DER [8].

OpenSSL codifica todas sus estructuras a DER antes de utilizarlas. Las similitudes entre *Xic* y *Xac* permiten reutilizar las funciones *Xic*, como hash, firma y verificación. Por tanto, solamente es necesario incluir las funciones de alto nivel.

Los siguientes ejemplos muestran el conjunto de funciones que son utilizadas para incluir información en los campos del certificado. El prefijo *X509AT* detalla la categoría de las funciones, y facilita la tarea a los desarrolladores a la hora de decidir las funciones apropiadas que deben utilizar.

```
X509AT_set_serialNumber(X509AT *x, ASN1_INTEGER *serial)
X509AT_set_notBeforeTime(X509AT *x, ASN1_GENERALIZEDTIME *tm)
X509AT_set_notAfterTime(X509AT *x, ASN1_GENERALIZEDTIME *tm)
X509AT_set_holder(X509AT *x, X509AT_HOLDER *holder)
```

**Tercera Fase: Adición de los elementos a la compilación** Una vez completada la segunda fase, obtendremos un nuevo directorio para el proceso de compilación. Este directorio contiene el código fuente de *Xac* que consta de cuatro archivos: definición, estructura del certificado, funciones de acceso a los elementos del certificado y funciones de alto nivel.

El proceso de compilación necesita la modificación de algunos archivos de configuración para, por ejemplo, incluir los nuevos archivos generados. El archivo *util/mkdef.pl* contiene la definición y el archivo *util/libeay.num* contiene las funciones de bajo y alto nivel *Xac*.

Finalmente, la compilación es realizada utilizando los métodos definidos en la librería, obteniendo la librería que proporciona las funciones *Xac*.

## 4. Middleware

Esta sección describe un prototipo de administración de privilegios que ha sido creado utilizando, como plataforma, la implementación de *Xac* detallada previamente en OpenSSL. La propuesta ITU-T establece cuatro modelo de PMI: General, Control, Delegación y Modelo de Roles. Nuestro prototipo pretende abarcar los cuatro modelos, prestando un mayor interés en el modelo de Delegación. Otra implementación de la propuesta ITU-T, como es el caso del proyecto PERMIS [1], se centra en el Modelo de Roles.

### 4.1. Componentes del Sistema

Como se ha hecho patente en las secciones anteriores, nuestro trabajo está basado en la propuesta ITU-T X.509, pero también está influenciado por el reciente informe emitido por European Telecommunications Standards Institute (ETSI) [14] que se centra en los requisitos necesarios para los certificados de atributos.

La base para implementar la delegación es la existencia inicial de un propietario del privilegio y las acciones que se establecen para que la entidad inicial transfiera a otra entidad el control sobre los privilegios. El propietario inicial

de los privilegios es denominado *Source of Authorization* (SOA) en la recomendación de la ITU-T y *Attribute Granting Authority* (AGA) en el informe del ETSI.

El informe del ETSI detalla los elementos que deben ser parte de un sistema de Autorización basado en certificados de atributos, facilitando un esquema de la implementación del sistema para la inclusión del concepto de delegación.

Nuestro sistema está basado en los elementos descritos en las siguientes secciones, teniendo en cuenta la propuesta de la ETSI.

**Autoridad de Atributos** La *Autoridad de Atributos (AA)* ha sido desarrollada como un servidor de transacciones, usando un esquema cliente-servidor basado en hebras. El servidor, que es la hebra principal, es la encargada de escuchar las conexiones clientes entrantes.

Cuando una petición es establecida, el cliente debe ser autentificado. El sistema soporta varios mecanismos de autenticación como el tradicional login-password, u otros más avanzados, como los basados en directorios Ldap, Biométricos, etc. Los mecanismos de autenticación pueden ser extendidos mediante un sistema de plugin desarrollado en el servidor.

Previo al proceso de autenticación, cliente y servidor deben estar de acuerdo en los métodos de autentificación. Si el proceso de autenticación ha sido fructífero, se crea una hebra hija dedicada a atender al correspondiente cliente.

Las funciones de una AA son relativas a la definición de la peticiones que aceptan. Las peticiones están compuestas de los elementos *Class*, *Function* y *Type*. Las siguientes sentencias son ejemplos de peticiones que una AA puede aceptar:

***CLASS-CERTIFICATE*:** Esta clase incluye las peticiones relativas a la creación, firma y publicación de certificados:

- *FUNCTION-GET-SIGNED* - Devuelve los certificados firmados por la AA.
- *FUNCTION-PUBLISH-LDAP* - Firma (si es necesario) y publica el certificado en LDAP

***CLASS-REVOCATION*:** Esta clase incluye las peticiones relativas a la revocación de certificados

- *FUNCTION-REVOKE* - Revoca un certificado usando su número de serie como referencia

***CLASS-CONFIGURATION*:** Esta clase incluye las peticiones relativas a la administración de la información de la AA

- *FUNCTION-ADD-ATTRIBUTE, FUNCTION-MODIFY-ATTRIBUTE y FUNCTION-REMOVE-ATTRIBUTE* - Administra las lista de atributos disponibles.
- *FUNCTION-ADD-EXTENSION y FUNCTION-REMOVE-EXTENSION* - Administra las lista de extensiones disponibles.

- *FUNCTION-ADD-RESTRICTION, FUNCTION-MODIFY-RESTRICTION, y FUNCTION-REMOVE-RESTRICTION* - Administra las listas de las restricciones disponibles.

**Operador de la Autoridad de Atributos** Esta sección presenta el módulo de comunicación multiplataforma, que facilita a los desarrolladores la implementación de la comunicación entre cliente y AA.

El módulo puede ser usado como parte de un programa no interactivo, o como parte un programa interactivo. En el caso interactivo, una interfaz visual ha sido añadida para facilitar las tareas de administración al Operador de la Autoridad de Atributos. La figura 1 muestra el interfaz gráfico usado para las acciones de configuración.

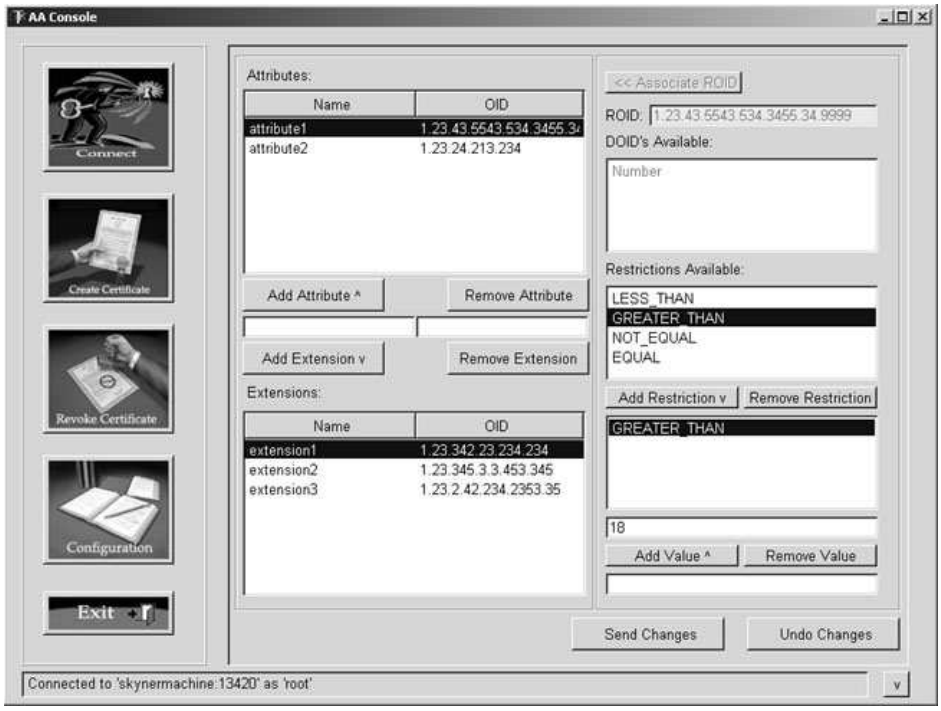


**Figura 1.** Captura de pantalla de la Autoridad de Atributos

**Cliente AA** El Cliente de AA envía las peticiones de tareas a realizar a la AA. Estas peticiones son convertidas en acciones que pueden realizar la AA si

el cliente tiene los permisos necesarios. La AA define y almacena los usuarios autorizados y las tareas permitidas para cada usuario autorizado.

Hay diferentes posibilidades de definir los permisos:

- ***CLASS*:** Permite todas las peticiones que incluye una clase como tipo de petición.
- ***CLASS* | *FUNCTION*:** Permite realizar solamente una función de la clase.
- ***PROFILE*:** Permite las acciones definidas en un determinado perfil.

*PROFILE* es definido como una combinación de permisos y acciones. Es posible que un usuario tenga cualquier combinación de permisos y perfiles.

Esta configuración permite establecer diferentes figuras en la infraestructura, cada una de ellas con un perfil específico. Por ejemplo, los siguientes roles están disponibles: Certificates Issuer, Certificate Revoker, Admin, etc.

Los siguientes ejemplos detallan posibles configuraciones de los perfiles:

- *PROFILE-CERTIFICATE-GENERATOR*: CLASS-CERTIFICATE | FUNCTION-GET-SIGNED CLASS-CERTIFICATE | FUNCTION-PUBLISH-LDAP
- *PROFILE-CERTIFICATE-REVOKE*: CLASS-REVOCATION | FUNCTION-REVOKE
- *PROFILE-MANAGER*: PROFILE-CERTIFICATE-GENERATOR | PROFILE-CERTIFICATE-REVOKE

Además, estos perfiles pueden ser asignados al Cliente AA:

- Manager: PROFILE-MANAGER
- Operator1: PROFILE-CERTIFICATE-GENERATOR
- Operator2: PROFILE-CERTIFICATE-GENERATOR
- Operator3: PROFILE-CERTIFICATE-REVOKE

El esquema presentado facilita la división de las tareas a los usuarios del sistema. Por ejemplo, si el cliente AA es un módulo de un servidor web, que requiere generar un certificado de atributo en el instante, es necesario declarar la siguiente función en la clase:

- CLASS-CERTIFICATE
  - FUNCTION-GET-SIGNED
  - FUNCTION-PUBLISH-LDAP
  - *FUNCTION-GET-WEB-CERTIFICATE*: Emite un certificado de atributos para cierto usuario con cinco minutos de validez.

Además, la AA añade un usuario web:

- WebServerUser: CLASS-CERTIFICATE | FUNCTION-GET-WEB-CERTIFICATE

**Utilizando LDAP para la publicación de los Certificados de Atributos** Los *Xac* no pueden ser tratados de la misma forma que los *Xic*. Normalmente un *Xac* debe de estar vinculado a un *Xic*, y una entidad puede tener un número ilimitado de *Xac*. Esta situación difiere de las ya conocidas como es el caso de *Xic*. Por otra parte, el único elemento que distingue un *Xac* de otro *Xac* es el número serie.

El número de serie podría ser usado para almacenar el *Xac* en el directorio; sin embargo, esta no es una solución apropiada para obtener el *Xac*, porque el número de serie no representa la información sobre el usuario y los atributos que almacena.

Por esta razón, el esquema usado para almacenar y obtener *Xac* en un Directorio es el definido en la propuesta [2], añadiendo un campo obligatorio (campo MUST) a la definición (attributeCertificateAttribute):

```
(1.2.826.0.1.3344810.1.0.16
   NAME 'x509AC'
   SUP x509base
   STRUCTURAL
   MUST (  x509version
           x509serialNumber
           x509validityNotBefore
           x509validityNotAfter
           attributeCertificateAttribute)

   MAY  (  x509acHolderPKCSerialNumber
           x509acHolderPKCissuerDN
           x509acHolderRfc822Name
           ...
   ... ...)
```

El esquema utilizado para almacenar y obtener la *lista de certificados de atributos revocado* (ACRL) es el especificado en el estándar [13]. En este caso, se adoptó la misma solución que *Xic*, debido a que ambos son revocados utilizando el número de serie.

## 5. Conclusión y Trabajo Futuro

El presente trabajo muestra ciertos elementos necesarios para construir un prototipo de infraestructura que proporcione soporte para la delegación de privilegios. La propuesta de la ITU-T y el informe reciente de ETSI han sido como utilizado como base para para desarrollar el prototipo.

En el momento de la publicación del artículo, solamente la librería IAIK daba soporte a *Xac*. IAIK está implementada en JAVA. Las modificaciones realizadas a OpenSSL añaden otra librería que implementa *Xac* en los lenguajes C y C++. La inclusión de la funcionalidad en la librería OpenSSL permite que la comunidad utilice el soporte de *Xac* usando una librería de código libre.

Actualmente, la única infraestructura de administración de privilegios (PMI) conocida es la realizada en el proyecto PERMIS. El proyecto se basa en la librería IAIK para la implementación de certificados de atributos. Este proyecto se centra en las infraestructuras de control de acceso basadas en roles que utilizan los certificados de atributos X.509 para almacenar los roles de los usuarios.

Estudios iniciales sobre el proyecto PERMIS nos revelaron que el *Xac* generado no cumple fielmente el estándar, concretamente introduce una ligera modificación en la estructura *Xac*; más precisamente en la entrada *digestAlgorithm* del campo *ObjectDigestInto*. Esta variación afecta a los campos del emisor y del titular del *Xac*. En contraposición, la solución que planteamos en el presente trabajo cumple completamente los estándares definido por la ITU-T y los correspondientes RFC definidos por el IETF.

## Referencias


1. D. Chadwick, .An X.509 Role-based Privilege Management Infrastructure", *Business Briefing: Global Infosecurity*, 2002
2. D. Chadwick, M. Sahalayev, "Internet X.509 Public Key Infrastructure LDAP Schema for X.509 Attribute Certificates", draft-ietf-pkix-ldap-ac-schema-00.txt.
3. C. Covell, M. Bell, .OpenCA Guides for 0.9.2+", http://www.openca.org
4. C. Ellison et al. "SPKI Certificate Theory", Request for Comments 2693, IETF SPKI Working Group, September 1999
5. S. Farrell, R. Housley, .An Internet Attribute Certificate Profile for Authorization", Request for Comments 3281, IETF PKIX Working Group, April 2002
6. ITU-T Recommendation X.509, "Information Technology - Open systems interconnection - The Directory: Authentication Framework", June 1997
7. ITU-T Recommendation X.509, "Information Technology - Open systems interconnection - The Directory: Public-key and attribute certificate frameworks", March 2000
8. B. Kaliski, .A Layman's Guide to a Subset of ASN.1, BER, and DER", November 1993
9. Tom Parker, Denis Pinkas,"Sesame v4 Overview", Issue 1. December 1995
10. J. Vollbrecht, P.Calhoun, S. Farrell, et al. RFC 2904: AAA Authorization Framework", August 2000
11. A. Sanin, "XML Security Library Tutorial", http://www.aleksey.com/xmlsec/
12. Mary R. Thompson, Abdelilah Essiari, Srilekha Mudumbai,Çertificate-based Authorization Policy in a PKI Environment", TISSEC 2003
13. M. Wahl,RFC 2256:A Summary of the X.500(96) User Schema for use with LDAPv3",December 1997
14. ETSI TS 102 158,.Electronic Signatures and Infrastructures (ESI); Policy requirements for Certification Service Providers issuing attribute certificates usable with Qualified certificates", V1.1.1, October 2003
15. http://www.apache-ssl.org/
16. http://www.openssh.com/
17. http://www.opera.com/
18. "Kerberos: The Network Authentication Protocol", http://web.mit.edu/kerberos/
19. http://www.columbia.edu/ ariel/ssleay/