Instituto Universitario Aeronáutico

Trabajo Final de Posgrado

Especialidad en Seguridad Informática

Sellos de Competencia (Certificados de Atributos) para la Firma Digital de la República Argentina

Ing. Gonzalo Argüello

Tutor: Esp. Ing. Fernando Boiero

Índice

1 Introducción

La Decisión Administrativa N° 927 del 30 de Octubre de 2014 de la Jefatura de Gabinete de Ministros (en adelante "*DA 927*") establece el marco normativo de Firma Digital de la República Argentina poniendo en consideración una larga lista de decretos y decisiones que ocurrieron desde la sanción de la Ley 25.506 del 14 de Noviembre de 2001 (en adelante "*Ley de Firma Digital*") y teniendo en cuenta el avance tecnológico ocurrido y la experiencia adquirida durante ese lapso de casi 13 años. En este sentido resuelve establecer una política única para todos los certificadores licenciados e incorporar a la infraestructura nuevos servicios de firma digital entre lo que figura la emisión de **Sellos de Competencia** como herramienta para la confirmación de roles tales como la condición de titularidad de matrícula profesional, cargos en distintas organizaciones o atribuciones de carácter similar.

Para la implementación de los mismos, dicha Decisión Administrativa propone el uso de **Certificados de Atributos x509** según RFC 5755: *An Internet Attribute Certificate Profile for Authorization*. Sin embargo, en la actualidad no se encuentra ampliamente difundido el uso de los mismos, menos aún con el propósito que pretende la Ley. Esto se debe en gran parte a que se trata de estándares con pocos años de vida, incluso en lo que a firma digital se refiere desde el punto de vista de la legislación en los diferentes países. Esto lo convierte en un tema de interés para la investigación ya que la propuesta de soluciones estaría a la vanguardia de la tecnología.

2 Objetivos y Alcance

2.1 Situación de oportunidad

Los nuevos servicios de firma digital propuestos por la Ley no registran actualmente implementaciones conocidas o difundidas entre quienes proveen dichos servicios. Si bien tampoco se ha determinado la entrada en vigencia de estos, nos encontramos en una etapa exploratoria para determinar la mejor forma de implementarlos.

De esta manera, la investigación de estándares y las alternativas tecnológicas constituye una oportunidad para proponerun primer acercamiento sobre cómo se podría, por ejemplo, emitir y hacer uso de un Sello de Competencia como lo indica la Ley.

2.2 Objeto de estudio

En el presente trabajo analizaremos los conceptos involucrados en la Infraestructura de Firma Digital y los estándares tecnológicos a los que adhiere para hacer una propuesta sobre cómo realizar la implementación concreta de un **Sello de Competencia** y hacer uso de él para firmar un documento digitalmente.

2.3 Objetivos y resultados esperados

2.3.1 Objetivo general

Implementar un **Sello de Competencia** adecuado a la normativa más reciente de la Infraestructura de Firma Digital de la República Argentina para firmar un documento electrónico.

2.3.2 Objetivos específicos

- Analizar e interpretar el estándar tecnológico propuesto en RFC 5755 para adecuar la DA 927.
- Emitir Sellos de Competencia con el perfil propuesto en el Anexo IV de la DA 927.
- Firmar digitalmente un documento utilizando el Sello de Competencia.
- Verificar que la firma contenga los atributos otorgados con el Sello de Competencia.
- Hacer uso de herramientas *open-source*.

2.3.3 Resultados esperados

Validar la firmar digital en un documento verificando los Sellos de Competencia con los que el firmante extendió su rúbrica.

2.4 Delimitación del proyecto

El presente trabajo se circunscribe ala emisión de un Sello de Competencia según el perfil indicado en el Anexo IV de la DA 927 y su uso para firmar digitalmente un documento electrónico. Para la generación, firma y verificación del mismo se desarrollará un software ad-hoc que pretende cumplir exclusivamente con los objetivos de este proyecto.

Adicionalmente, será necesario emitir tres certificados digitales que servirán para ejemplificar una cadena de confianza y formalizar el esquema necesario que prevén tanto los estándares tecnológicos como la Ley. Estos serán generados con una herramienta a especificar en el desarrollo del trabajo y tienen validez sólo para el caso en cuestión.

3 Marco teórico de una infraestructura de clave pública

Una PKI (Public Key Infrastructure o Infraestructura de Clave Pública) es una conjunción, no sólo de elementos de hardware y software, sino también de políticas y procedimientos de seguridad necesarios que permiten la ejecución de operaciones criptográficas como el cifrado, la Firma Digital o el no repudio de los mensajes electrónicos. Es común observar la banalización o simplificación del término PKI, siendo éste utilizado sólo para referirse al uso de algoritmos de clave pública en comunicaciones electrónicas. Este significado es incorrecto ya que no se requieren elementos propios de una PKI para usar algoritmos de clave pública.

En general, una PKI permite a dos partes disponer de confidencialidad, autenticación e integridad en las comunicaciones sin tener que compartir ninguna información de antemano.

La criptografía de clave pública o criptografía asimétrica se basa en la generación de 2 claves, una pública y una privada, donde la primera es entregada libremente y la segunda es mantenida en total aislamiento para que nadie más que su generador pueda accederla o conocerla. A su vez, igualmente vital es que los mecanismos criptográficos (algoritmos) garanticen que la generación de claves sea única, es decir, que la pareja de claves se pueda generar una única vez para asegurar que nadie más pueda obtener casualmente la misma pareja de claves.

Los algoritmos de clave pública (por ejemplo RSA [01]), se basan en que cada usuario utiliza un par de claves relacionadas matemáticamente, de tal forma que una de ellas descifra el cifrado que se realiza con la otra. Estos algoritmos tienen la propiedad adicional de que, conociendo una de las claves del par, es computacionalmente imposible deducir la otra. Una de estas claves, que debe permanecer siempre en secreto, se conoce como privada y la otra como pública. Estos algoritmos permiten realizar dos operaciones:

- **Cifrado:** Un mensaje cifrado con la clave pública de un destinatario no puede ser descifrado por nadie excepto por el destinatario que está en posesión de la correspondiente clave privada. Este mecanismo proporciona confidencialidad.
- **Firma digital:** Un mensaje cifrado con la clave privada del emisor puede ser descifrado por cualquiera que tenga la clave pública de dicho emisor, probando de esa manera que sólo ese emisor pudo cifrar el mensaje y que no ha sido modificado. La Firma Digital proporciona autenticación.

Para utilizar los algoritmos de clave pública, cada una de las dos entidades que desean intercambiar información deben disponer de un par de claves relacionadas matemáticamente: una privada, que sólo conoce su propietario y otra pública que debe ser conocida por cualquiera que desee comunicarse con él.

La fortaleza de los algoritmos de clave pública más utilizados, reside en la dificultad de factorizar números grandes ya que la generación de claves se basa en elegir dos números primos *pseudoaleatorios* [02]. En la figura siguiente se observa el esquema genérico de generación de claves para un usuario.

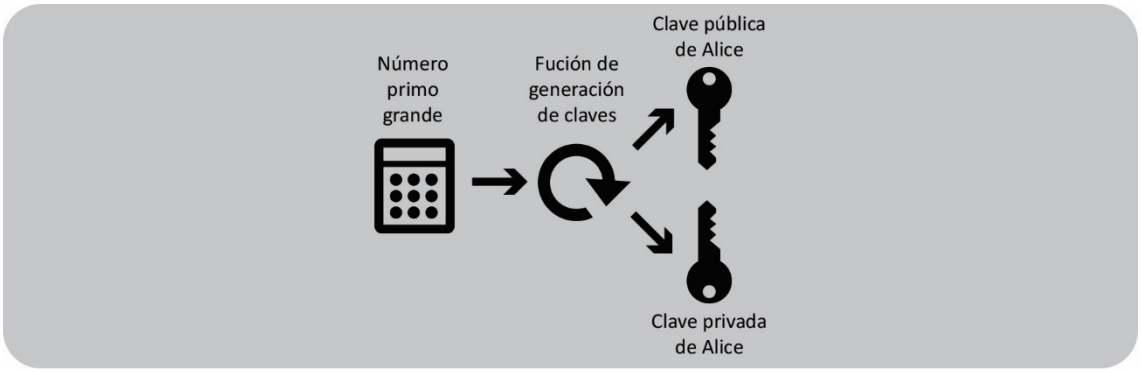

Ilustración 1 - Generación de claves pública y privada

Habiendo analizado la generación de claves, a continuación se describirán dos usos comunes en la infraestructura de clave pública.

3.1 Cifrado de datos con algoritmos de clave pública

El cifrado de datos con algoritmos de clave pública se realiza de la siguiente forma: suponiendo que Bob desea enviarle un mensaje a Alice; Bob tomará el mensaje que desea enviar y lo cifrará con la clave pública de Alice; cuando Alice reciba el mensaje utilizará su clave privada para descifrarlo y leer el mensaje original que Bob deseaba enviarle. En la **Ilustración 2 - Cifrado con Clave PúblicaIlustración 2** se puede ver cómo funciona este mecanismo. Como Alice es la única que conoce su propia clave privada, sólo ella podrá descifrar el mensaje que le envía Bob. Mediante este procedimiento conseguimos realizar envío de datos con confidencialidad.

Debido a que los algoritmos asimétricos son bastante más lentos que los simétricos, lo que se hace realmente en este tipo de cifrados es:

1. Sortear una clave para un algoritmo de cifrado simétrico y cifrar el mensaje que se desea enviar.
2. Cifrar esta clave simétrica con la clave pública del destinatario y enviar el resultado junto con el cifrado anterior.
3. Cuando el destinatario reciba el mensaje descifrará la clave simétrica con su clave privada y luego la utilizará para descifrar el mensaje original.

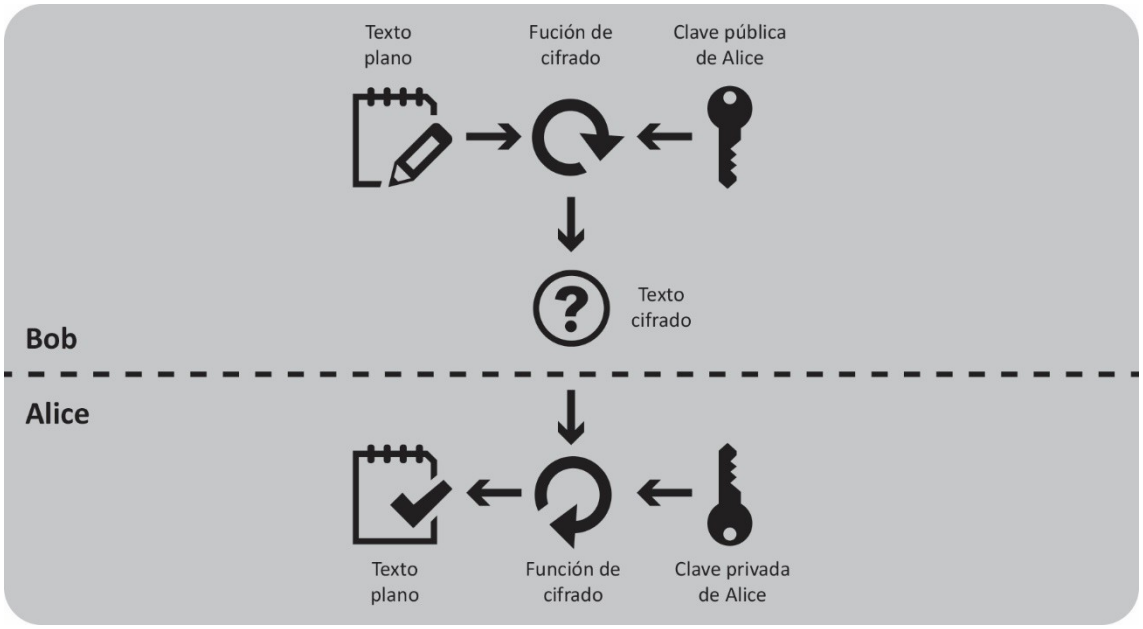

Ilustración 2 - Cifrado con Clave Pública

3.2 Firma Digital

La Firma Digital es un procedimiento que permite simular la seguridad que proporciona la firma manuscrita convencional. Como se puede ver en la **Ilustración 3**, realizar una Firma Digital consiste en lo siguiente, planteando que Alice desea firmar digitalmente un documento para enviar a Bob:

1. Alice tomará el documento y le aplicará una función de hash para generar un resumen del mismo.
2. Luego cifrará ese resumen con su clave privada para producir la **Firma Digital**.
3. Alice enviará a Bob el documento original y la **Firma Digital**.
4. Bob, por un lado, aplicará al documento original la misma función de hash y, por otro, descifrará con la clave pública de Alice la Firma Digital.
5. Por último, Bob comparará ambos resultados para verificar la firma.

Con este mecanismo se obtiene autenticación del emisor.

Como hemos visto en el punto anterior, los algoritmos asimétricos son lentos y cifrar un documento grande puede tener un costo computacional elevado. Por este motivo lo que se hace en lugar de cifrar todo el mensaje es aplicar una “función hash” al mensaje original y cifrar el resultado.

Una función hash es una transformación que toma como entrada una secuencia de longitud arbitraria y devuelve una secuencia de longitud fija que se denomina valor hash o resumen. El hash es un tipo de "huella digital" del documento original. Una función hash debe tener las siguientes propiedades:

- La función debe ser no invertible: dado un valor hash h debe ser computacionalmente imposible encontrar un valor m tal que *h = hash(m)*.
- Dado un valor m1 debe ser difícil encontrar un valor distinto m2 tal que *hash(m1) = hash(m2)*.
- Debe ser difícil encontrar dos valores m1 y m2 tal que *hash(m1) = hash(m2)*.

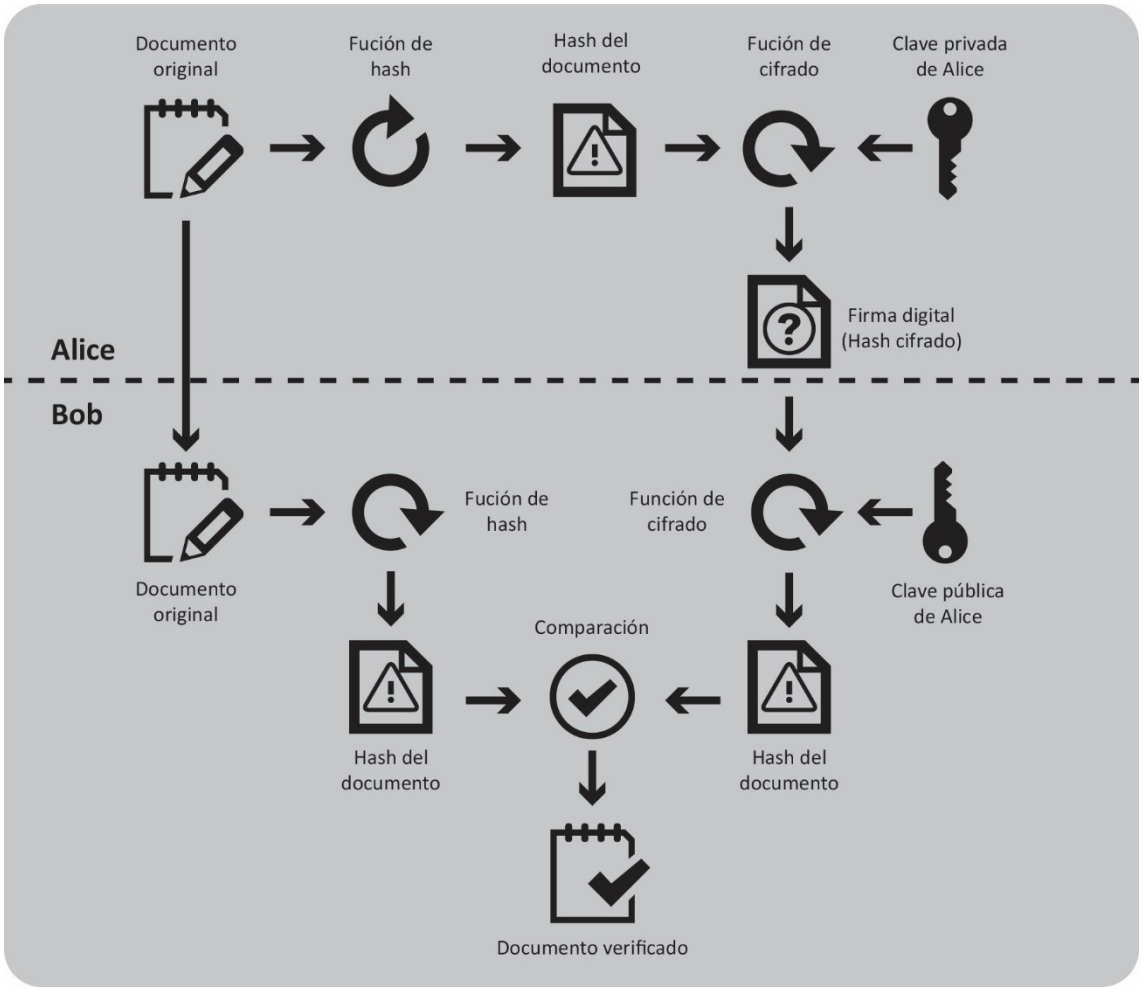

Ilustración 3- Firma Digital

Un componente esencial de una infraestructura de clave pública es el certificado digital. Cada certificado contiene la clave pública acompañada por otros parámetros relacionados con la clave. En el proceso de firmado digital se utiliza el certificado digital para constatar que la clave pública, necesaria para la verificación de la firma, pertenece al firmante. Adicionalmente, cada certificado contiene un conjunto de datos relevantes que son definidos en la recomendación ITU-TX.509 por medio de una estructura de certificado que actualmente se encuentra en su versión 3 [03]. En la sección siguiente se detallan sus características y atributos.

3.2.1 Certificados digitales

Los certificados digitales son los elementos basales de una infraestructura de clave pública. La finalidad de los certificados digitales es que mediante un documento firmado se pueda asociar la clave pública con una identidad inequívoca. Los datos que pueden ser detallados en la identidad son, por ejemplo, el nombre de la persona o de la organización, su país de procedencia, su dirección de e-mail, entre otros. Un certificado digital contiene los siguientes elementos fundamentales:

- La clave pública del sujeto.
- Los datos de identificación del sujeto.
- Firma Digital de una tercera parte que asegura que los dos elementos anteriores están relacionados.
- El sujeto asociado al certificado (persona o dispositivo).

- La Autoridad de Certificación (AC) que emite.
- La clave pública del par de claves.
- Los algoritmos usados en el certificado.
- Información de validez y determinación de la revocación.
- Distintas extensiones de X.509 versión 3.

En una PKI la firma del certificado la realiza una tercera parte confiable (Autoridad de Certificación), que es la que da fe acerca de la relación entre una identidad real y una electrónica. En otros sistemas, como los que siguen el estándar OpenPGP [04], se utilizan los llamados esquemas de confianza, en los que son unos usuarios los que certifican la identidad de otros, en lugar de depender de una única entidad confiable.

Respecto de extensiones, se destaca que proveen métodos para asociar atributos adicionales con usuario o claves públicas y para manejar la jerarquía de certificación. Es posible también definir extensiones privadas para completar información específica que sea de utilidad para los usuarios de los certificados. Dentro de un certificado, las extensiones se dividen en críticas y no críticas, actuando las primeras como causal de rechazo del certificado si no fueren reconocidas.

De acuerdo con X.509 v3 (RFC3280) [05], las extensiones estándares son:

Nombre de la extensión	**Descripción**
Authority Key Identifier	Provee los medios para identificar la clave pública correspondiente a la clave privada usada para firmar el certificado.
Subject Key Identifier	Identifica la calve pública certificada por este certificado. Provee una forma de distinguir claves públicas si más de 1 está disponible para cierto nombre de sujeto.
Key Usage	Define el propósito de la clave contenida en el certificado. Usos posibles: • digitalSignature • nonRepudiation • keyEncipherment • dataEncipherment • keyAgreement • keyCertSign • cRLSign • encipherOnly • decipherOnly
Private Key Usage Period	Permite al generador del certificado especificar un período de validez diferente para la clave privada que para el certificado en sí.
Certificate Policies	Permiten informar los términos de las políticas bajo las cuales el certificado fue generado y los propósitos de su uso.
Policy Mappings	Se utiliza sólo en certificados de Autoridades de

	Certificación. Puede ser útil en el escenario de pares cruzados de certificados.
Subject Alternative Name	Permite adosar identidades adicionales al sujeto del certificado.
Issuer Alternative Names	Se utiliza para asociar la identidad en el formato usado en Internet con el generador del certificado.
Subject Directory Attributes	Permite detallar atributos de identificación del sujeto .
Basic Constraints	Indica si el sujeto del certificado es una Autoridad de Certificación
Name Constraints	Sólo se puede utilizar en certificados de Autoridades de Certificación. Define un espacio de nombres dentro del cual todos los nombres de los sujetos en certificados subsecuentes en la ruta de certificación deben localizarse.
Policy Constraints	Sólo se puede utilizar en certificados de Autoridades de Certificación. Restringe la validación de ruta de certificación a 2 vías.
Extended Key Usage	Da la posibilidad de ampliar los propósitos iniciales que se le dieron con la extensión Key Usage.
CRL Distribution Points	Define cómo se conseguirá la información de listas de certificados revocados.
Inhibit Any-Policy	Sólo se puede utilizar en certificados de Autoridades de Certificación. Con determinados valores, permite que no coincida con los criterios para determinadas políticas.
Freshest CRL (a.k.a. Delta CRL Distribution Point)	Provee un puntero a la lista de certificados revocados más actualizada posible.

Los tipos de archivo y por ende sus extensiones de archivo de certificados X.509 son:

- **.CER:** Certificado codificado en CER [06], algunas veces es una secuencia de certificados.
- **.DER:** Certificado codificado en DER [06].
- **.PEM:** Certificado codificado en Base64, encerrado entre "-----BEGIN CERTIFICATE-----" y "-----END CERTIFICATE-----". Un archivo .PEM puede contener certificados o claves privadas, encerrados entre las líneas BEGIN/END apropiadas.
- **.P7B:** Es un estándar para firmar o cifrar datos. Dado que el certificado es necesario para verificar datos firmados, es posible incluirlos en la estructura *SignedData*.
- **.P7C:** Estructura PKCS#7 *SignedData* sin datos, sólo certificado(s) o CRL(s).
- **.PFX:** Personal inFormation eXchange.

- **.P12:** PKCS#12, puede contener certificado(s) (público) y claves privadas (protegido con clave). Evolucionó del estándar PFX y se usa para intercambiar objetos públicos y privados dentro de un archivo.

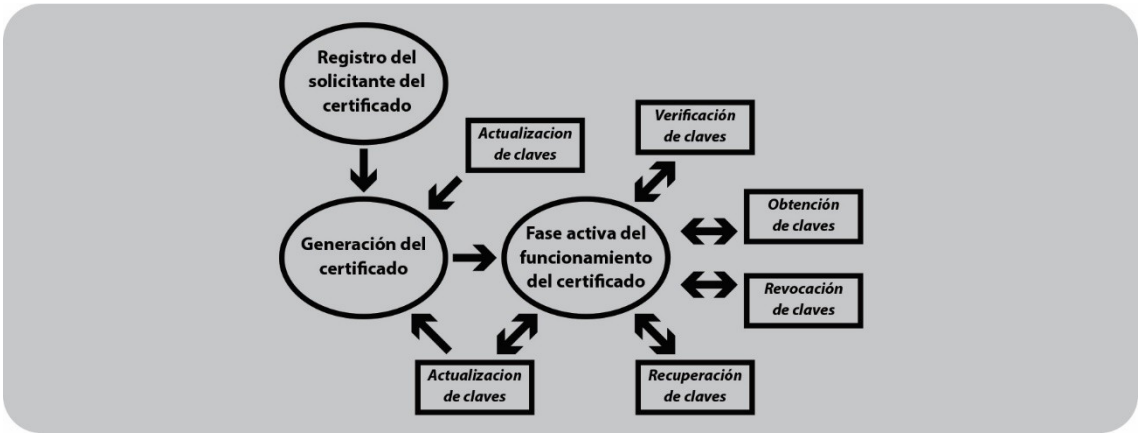

Ilustración 4 - Ciclo de vida de un Certificado Digital

Es relevante destacar que los certificados digitales tienen un ciclo de vida con las siguientes etapas [07], según los diferentes eventos o hitos que se sucedan durante su utilización: a) Revocación; b) Expiración; c) Renovación; d) Reemisión del par claves del usuario; e) Actualización de datos del certificado. Se esquematiza el ciclo de vida de los certificados digitales en la**Ilustración 4.**

3.2.2 Propiedades de seguridad que brinda una PKI

Las características de Seguridad de la Información que pueden obtenerse al implementar una infraestructura PKI son las que se enuncian a continuación:

- **Confidencialidad:** se refiere a la capacidad de proteger a una comunicación entre dos interlocutores para no ser interceptada ni interferida por una tercera persona que no forme parte de la comunicación. Para ello, la PKI utiliza mecanismos de cifrado que evitarían que individuos no autorizados se hagan de los mensajes.
- **Autenticación:** apunta a que se brinde acceso a una comunicación electrónica sólo a quienes se pueda corroborar su identidad y se pueda constatar su habilitación. En este caso, los certificados digitales y la estructura de confianza intrínseca a la PKI conforman una alternativa de autenticación a la tradicional presentación de credenciales (usuario/contraseña).
- **Integridad:** es necesario comprobar que la información enviada sea exactamente la misma que la que se reciba. El control apunta a garantizar que los datos no fueron alterados dentro del canal de comunicación. Adicionalmente, el control sirve para corroborar que un archivo se mantiene inalterado en el tiempo o en distintos medios de almacenamiento, garantizando que posee el mismo contenido. A nivel técnico, este control es logrado a través de las funciones de hash que permiten efectuar comparaciones inequívocas de integridad de datos.
- **No-repudio:** esencialmente implica que los interlocutores no pueden negar haber enviado los mensajes que los muestren como remitentes. Esta propiedad hace que, ante algún problema entre las partes al haber intercambiado mensajes electrónicos, sea innegable evidencia presente dentro del sistema de comunicación que pueda ser utilizada para probar con suficiente certeza lo que realmente sucedió. Esto cobra

especial importancia en operaciones financieras o trámites de índole legal cuyas consecuencias pueden ser de relevancia.

3.2.3 Elementos constituyentes de una PKI

3.2.3.1 Autoridad de Certificación

Como ya se mencionó previamente, el objetivo de una PKI es generar certificados digitales que asocien la identidad real de una persona, organización o incluso un dispositivo hardware con una clave pública. Una vez que una entidad dispone de un par de claves necesita que una tercera parte certifique y asocie ese par de claves con su identidad ante otras entidades. Esta tercera parte confiable en el mundo de las *PKIs* se denomina Autoridad de Certificación.

Una Autoridad de Certificación es un agente encargado de generar, custodiar y utilizar su propia identidad digital y de emitir los documentos digitales firmados que constituyen los certificados que emite. Un certificado digital es, en esencia, un documento digital firmado, cuya estructura es públicamente conocida y que incluye los datos verificados necesarios para poder afirmar lo que ese documento certifica.

Hasta este punto se disponían de mecanismos para realizar comunicaciones autenticadas y confidenciales, pero existía el problema de no tener certeza de que la clave pública de la otra entidad correspondiera a quién se creía, a menos que se verificara de manera externa. Ahora en lugar de tener la clave pública de otra entidad, disponemos de un certificado digital que asegura que dicha entidad es quien dice ser siempre y confiamos en la Autoridad de Certificación que emitió su certificado.

Existen dos tipos de Autoridades de Certificación en función de quién firma su propio certificado:

Autoridad de Certificación Raíz: Son aquellas que firman sus propios certificados con su clave privada. Este tipo de Autoridades de Certificación no disponen de una tercera parte confiable que aseguren que son quienes dicen ser. Es cada una de las entidades que vayan a utilizar el certificado de dicha autoridad la que debe decidir si confía o no en él mismo. La confianza o no en este tipo de autoridades suele venir dada por su reputación o conocimiento personal, por ejemplo, algunas grandes empresas dedicadas al negocio de las PKIs disponen de Autoridades de Certificación raíz generalmente aceptadas, e incluso instaladas como confiables por defecto en muchos navegadores web, sistemas operativos y demás software.

Autoridades de Certificación Subordinadas: Este tipo de Autoridad de Certificación dispone de un certificado que no está firmado por sí mismas, sino por otra Autoridad de Certificación, ya sea raíz o no.

Con estos dos tipos de Autoridades de Certificación vemos que se pueden crear estructuras jerárquicas arborescentes en las que unas Autoridades de Certificación pueden firmar el certificado de otra autoridad o bien de un usuario final.

3.2.3.2 Autoridad de Registro

Este elemento de la PKI es el encargado de recibir las solicitudes de emisión de certificados para una determinada Autoridad de Certificación. Estos agentes deben controlar que todos los datos y documentos (digitales o no) que aporta una entidad son auténticos y que permiten identificar inequívocamente al solicitante del certificado.

Asimismo, tienen la responsabilidad de decidir si la solicitud es pertinente o no, y si puede emitirse el correspondiente certificado. Las Autoridades de Registro son una parte

fundamental de una PKI ya que son las que comprueban la relación entre una clave pública y su propietario antes de que la Autoridad de Certificación la plasme en un certificado digital.

3.2.3.3 Dispositivos de usuario

Para que la utilización de los certificados sea practicable para los usuarios, se deben disponer herramientas software y/o hardware que actúen en su nombre y les permitan hacer uso de sus identidades digitales, más específicamente que sean capaces de:

- **Generar identidades digitales:** Los dispositivos deben contener un generador de secuencias aleatorias que les permitan generar el material de partida de las claves que constituyan una identidad digital, así como almacenarlas de forma segura.
- **Solicitar certificados digitales:** Una vez autorizado por la Autoridad de Registro, el dispositivo debe permitir solicitar a la Autoridad de Certificación un certificado digital asociado a un par de claves.
- **Realizar operaciones criptográficas:** Debe permitir firmar digitalmente o cifrar datos, con sus identidades digitales que son objetivos funcionales una PKI.

3.2.4 Objetivos y finalidades de una PKI

3.2.4.1 Verificación de certificados

Habiendo ya descrito los elementos de una PKI y la forma en que se generan los certificados digitales se debe analizar su modo de empleo. En este punto se sabe que en lugar de disponer simplemente de la clave pública de otra entidad, disponemos de su certificado.

Antes de utilizar un certificado es necesario verificar que está firmado por una Autoridad de Certificación confiable para el usuario. Para ello se deben realizar los siguientes pasos:

1. Si el certificado está auto firmado, es decir, la clave pública está firmada con la clave privada del mismo par de claves, entonces el usuario debe decidir si confía en el certificado o no.
2. Si el certificado está firmado por una Autoridad de Certificación raíz, el certificado será confiable si la Autoridad de Certificación que lo firmó es confiable para el usuario.
3. Si el certificado está firmado por una Autoridad de Certificación no raíz, entonces: si la Autoridad de Certificación es confiable, el certificado lo será y si no se debe verificar el certificado de dicha autoridad desde el paso 2. A este procedimiento se le denomina verificación de la cadena de certificación.

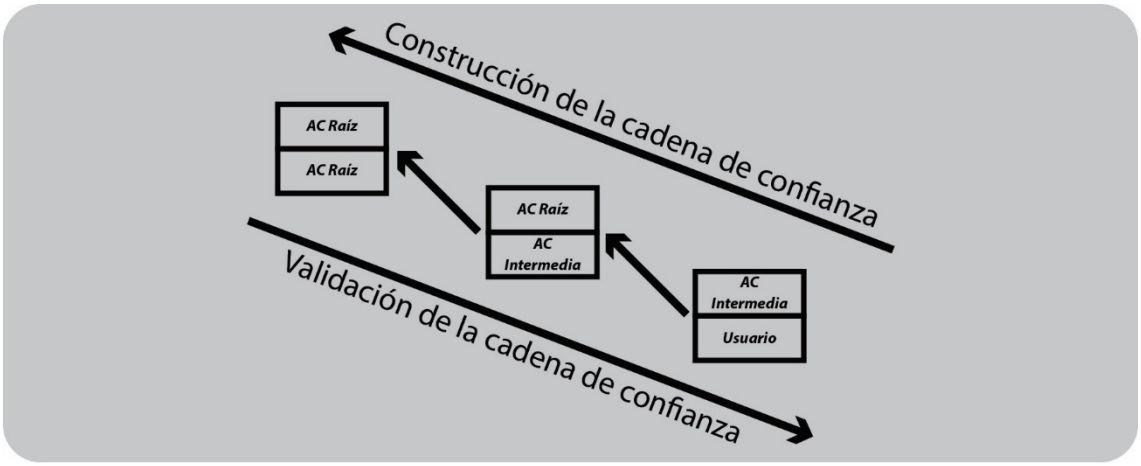

Ilustración 5 - Construcción de la cadena de confianza

Viendo el procedimiento de verificación de certificados queda clara la estructura jerárquica de las PKIs.

3.2.4.2 Gestión de certificados

Precedentemente se detalló el proceso a seguir para solicitar un certificado digital, pero no es el único servicio que debe proporcionar una PKI. Los certificados digitales tienen una fecha de validez, que impone la Autoridad de Certificación, durante la que se compromete a certificar que un sujeto está vinculado a un par de claves. Una PKI debe disponer de protocolos para realizar otro tipo de trámites distintos de la solicitud de certificados entre los que se encuentran:

- **Revocación:** Un usuario o Autoridad de Certificación puede querer rescindir la asociación entre un individuo y su par de claves. Este proceso se conoce como revocación de certificados. La necesidad de revocar un certificado puede deberse a varios factores, por ejemplo, un compromiso de seguridad de la clave privada por parte de un usuario o el cese de afiliación de un individuo dentro de la organización reflejada en su certificado.
- **Recertificación:** Los certificados digitales tienen fechas de validez que indican durante qué período un certificado es válido. Una vez finalizado el plazo de validez de un certificado, puede resultar útil que éste se pueda renovar durante un nuevo período de tiempo siempre que los datos de identificación del usuario y su par de claves sigan siendo válidos.
- **Renovación:** La renovación de un certificado consiste en solicitar un nuevo certificado con los mismos datos que el actual pero para un nuevo par de claves. Este procedimiento se realiza cuando un usuario desea cambiar el par de claves que utiliza, pero todos los datos de identificación siguen siendo válidos. La recertificación de certificados se puede ver como un caso particular de la renovación, donde el usuario mantiene el mismo par de claves.

La revocación de certificados debe ser un servicio obligatorio que debe proporcionar toda PKI, mientras que la recertificación y la renovación son opcionales. Algunas PKI pueden, además, obligar al usuario a volver a pasar por la Autoridad de Registro para volver a demostrar su identidad antes de permitir la realización de operaciones de recertificación y renovación. Todas las reglas que se deben cumplir para realizar estas operaciones deben ser mantenidas en la Política de Certificación que debe disponer toda Autoridad de Certificación y que debe estar disponible para todos sus usuarios.

3.2.4.3 Comprobación del estado de un certificado

En el punto precedente se indicó que algunos certificados pueden no ser válidos aunque estén dentro de su período de validez, ya que alguna circunstancia extraordinaria pudo provocar que se revocaran. Esto hace necesario disponer de mecanismos adicionales que permitan comprobar si un certificado ha sido revocado, y por tanto ya no es confiable lo que certifica, o no. La manera de acceder a este tipo de servicios suele venir definida en el propio certificado digital que se desea verificar.

Existen dos tipos de servicios que permiten realizar estas comprobaciones:

3.2.4.3.1 Servicios offline

Las Autoridades de Certificación pueden poner a disposición de los usuarios, cada cierto tiempo, una lista con los certificados que están dentro de su período de validez pero que han

sido revocados. Es tarea de los dispositivos de usuario obtener una copia actualizada de estas listas periódicamente para asegurarse de que un certificado no ha sido revocado antes de utilizarlo.

Para este tipo de servicio se utiliza el método de *Certificate Revocation List* (CRL), el cual es una lista con sello de tiempo, firmada por una Autoridad de Certificación, que contiene los números de serie de los certificados que han sido revocados y que puede estar en un repositorio público a disposición de los usuarios.

Cuando una aplicación requiere un certificado, además de comprobar la firma y la validez del mismo, debe controlar que el número de serie del certificado no esté presente en la última CRL emitida. Las listas de revocación de certificados se deben generar cada cierto tiempo, por ejemplo, cada hora o cada día. Cuando se revoca un certificado, éste deberá aparecer siempre en la siguiente CRL que se emita. Además, un certificado revocado deberá permanecer siempre en todas las CRLs siguientes hasta que se emita una CRL planificada para un momento posterior a la finalización de la validez del certificado. Esto evita el problema de que se revoque un certificado y que la próxima CRL se vaya a emitir después de la finalización de la validez del certificado; en este caso existiría un certificado revocado que nunca habría aparecido en una CRL. La mayor ventaja de este método de revocación es que las CRLs se pueden distribuir mediante los mismos medios que los certificados digitales, es decir, utilizando métodos de comunicación no confiables. La mayor limitación en el uso de CRLs es que la granularidad de la revocación en el tiempo está limitada por el período de publicación de las CRLs. Por ejemplo, si se realiza una revocación en este preciso momento, la información sobre esa revocación no estará disponible hasta que se publique la siguiente CRL, que puede ser dentro de una hora, un día o una semana dependiendo de la frecuencia de actualización de las CRLs.

3.2.4.3.2 Servicios online

Algunas PKIs disponen de servicios en línea que permiten a un usuario consultar en cualquier momento el estado de un certificado que va a ser utilizado.

El protocolo de comprobación de estado de certificados que se utiliza para este caso es el *Online Certificate Status Protocol* (OCSP).

Este protocolo provee un método para determinar el estado de revocación de un certificado X.509 sin necesidad de utilizar CRLs. A diferencia de las CRLs, el protocolo OCSP proporciona información sobre el estado de revocación de un certificado en el momento actual. Los clientes OCSP realizan consultas a los servidores OCSP y aplazan el uso del certificado hasta que reciben una respuesta sobre el estado del mismo. Cuando un servidor OCSP recibe una solicitud deberá comprobar que esté bien construida, que sea capaz de realizar el tipo de solicitud que le indica el cliente y que la información que necesita para procesar la solicitud es correcta. Todas las respuestas OCSP tienen que estar obligatoriamente firmadas digitalmente por la Autoridad de Certificación que emitió el certificado en cuestión o por una entidad confiable para el usuario para firmar respuestas OCSP. Este método es uno de los más utilizados en la actualidad para realizar consultas sobre el estado de los certificados. El diseño de este protocolo se realizó basándose en la información disponible en las CRLs y con el objetivo de que ambos mecanismos fueran totalmente compatibles.

3.2.4.4 Publicación de certificados

Una vez que un usuario dispone de un certificado digital le surge la necesidad de compartirlo con otras entidades para poder comunicarse de manera segura con ellas. Suponiendo que un

usuario desea enviar un mensaje cifrado a otro. Para ello el usuario debe disponer del certificado digital del destinatario. Con lo detallado hasta ahora el emisor debería solicitar al destinatario su certificado digital de manera convencional y después enviarle el mensaje cifrado que deseaba. Para evitar este inconveniente, las PKIs suelen proporcionar servicios de publicación de certificados, en los que los usuarios de una PKI, o incluso a veces cualquiera que lo necesite, pueda obtener los certificados emitidos para un usuario. Se reseñan a continuación dos métodos:

3.2.4.4.1 Publicación Web

Una forma simple de poner a disposición de los usuarios de una PKI los certificados de los demás miembros es utilizar un servidor Web de donde se puedan descargar mediante el protocolo HTTP. El acceso a este servidor puede ser público (para cualquier persona) o privado, introduciendo algún mecanismo de control de acceso, por ejemplo algún método simple como autenticación mediante usuario/contraseña, o algo más complejo mediante previa identificación con el certificado del usuario.

Una vez que se tiene acceso a este repositorio se puede mostrar al usuario una lista con los certificados disponibles para que descargue el que necesite o bien disponer de una interfaz Web que permita realizar búsquedas.

Algunas soluciones existentes, como EJBCA, disponen de este mecanismo para realizar la publicación de certificados. EJBCA proporciona dos páginas web, una con enlaces directos a los certificados de las Autoridades de Certificación en diferentes formatos y otra con una caja de texto en la que se pueden realizar búsquedas de certificados por el número de serie o el DN (*Distinguished Name*).

Esta solución no es muy buena ya que no está estandarizada, sino que cada aplicación la implementa de acuerdo a sus necesidades y por tanto no se integra directamente con las aplicaciones en las que se suelen usar los certificados.

3.2.4.4.2 Lightweight Directory Access Protocol (LDAP)

Este protocolo pertenece a la capa de Aplicación del modelo TCP/IP [08] y está basado en el estándar X.500 [09], teniendo como función la realización de consultas y modificaciones sobre un servicio de directorio, entendiéndose este último como un conjunto de objetos con atributos organizados en una jerarquía. Los servidores LDAP suelen utilizarse, para almacenar credenciales (usuarios y contraseñas) y también otro tipo de información como datos de contacto del usuario, certificados, etc. En resumen, el protocolo LDAP brinda acceso a un conjunto de información sobre una red y sus usuarios, cuya estructura de objetos y atributos es jerárquica y cuya raíz de la jerarquía se encuentra el “nodo raíz”.

3.3 Certificados de Atributos x509 (RFC 5755)

Los Certificados de Claves Pública x509 (en adelante “CCP”) que normalmente se gestionan con una PKI relacionan una identidad (persona física, persona jurídica, aplicación o sitio seguro) con una clave pública. Un Certificado de Atributos[10] (en adelante “CAtr”) o, como los nombra la DA 927: **Sello de Competencia**, es una estructura similar a la de un CCP sólo que no contiene clave pública. Un CAtr puede contener atributos que especifican la pertenencia a un grupo, un rol u otra información de autorización relacionada con su titular.

Para entender la diferencia y simplificar el concepto podemos usar la siguiente analogía. Un CCP es como un pasaporte: identifica al poseedor, tiende a durar mucho tiempo y su obtención no es tan simple. Un CAtr, en cambio, haría las veces de una visa de entrada: suele emitirla una

autoridad diferente, su validez está determinada por el tiempo en que la persona tiene autorización para ingresar a determinado país y para obtenerla primero debemos contar con nuestro pasaporte.

Si bien es posible incorporar información de autorización a un CCP mediante el uso de *extensiones*, esta práctica no es recomendada por dos motivos. En primer lugar, el tiempo de vida de una autorización no suele ser el mismo que el de una identidad. En segundo, quien emite un CCP no suele tener la autoridad para dar un permiso a alguien. Volviendo a la analogía, el país que emite un pasaporte para uno de sus ciudadanos no tiene la autoridad para darle una visa de entrada a otro país.

Los Certificados de Atributos x509 están pensados para proveer varios servicios de seguridad incluyendo control de acceso, autenticación del origen de datos y no repudio. Desde el punto de vista de la Firma Digital de la República Argentina es razonable pensar que se necesitan de estos 3 mecanismos de manera simultánea, por lo cual los CAtr deben estar relacionados con el CCP de su titular de manera que, a la hora de ser usados, se pueda validar que la persona está autorizada a firmar, sea quien dice ser y no pueda negar el acto de firma.

3.3.1 Distribución

En cuanto a la presentación de los CAtr, existen dos mecanismos. Por un lado el poseedor puede ofrecerlo al momento en que se le solicita demostrar, por parte del sistema verificador,si tiene ciertos permisos*("Push")*. Esta forma es conveniente en esquemas en donde los permisos del cliente se asignan dentro de su dominio de origen. La operación de autorización es más simple ya que el cliente presente "todo lo que el servidor necesita saber" y no se deben realizar conexiones adicionales entre servidor y cliente o entre el servidor y el emisor del CAtr.En otros casos es más conveniente que el cliente se autentique en el servidor y luego el servidor solicite ("*Pull*") el CAtr al cliente o a un repositorio. Este modelo es conveniente en esquemas en donde los permisos del cliente se asignan dentro del dominio de quien verifica la autorización. La Ley Argentina no indica qué método utilizar ante determinado caso por lo que la implementación de cada proveedor de servicios establecerá cuál alternativa conviene según se trate de firmar un documento digitalmente o utilizar el Sello como una credencial de autorización en algún sistema informático.

3.3.2 Perfil de un Certificado de Atributos

Un Certificado de Atributos es una estructura X.509 que está formado por tres partes principales, las cuales desglosaremos y ampliaremos:

- Información del Certificado de Atributos (*acinfo*)
- Algoritmo de firma (*signatureAlgorithm*)
- Firma del certificado (*signatureValue*)

3.3.2.1 Información del Certificado de Atributos (acinfo)

3.3.2.1.1 Versión (*versión*)

Sólo puede tener el valor *v2* para cumplir con esta norma.

3.3.2.1.2 Titular/Portador (*holder*)

Existen tres opciones para identificar al titular ("tenedor") de un CAtr. Para ello se sugiere elegir sólo uno de los siguientes campos:

- Identificador del Certificado base (*baseCertificateID*): debe indicar el número de serie y emisor del CCP del portador.

- Nombre de la entidad (*entityName*): debe contener exactamente lo mismo que figura en el campo *subject* del CCP.
- Hash de un objecto (*objectDigestInfo*): hash de un objeto que identifique directamente al portador. Por ejemplo, un ejecutable.

3.3.2.1.3 Emisor (*issuer*)

Para la versión 2 se debe completar el campo *issuerName* con el nombre distintivo del emisor tal como figura en elCertificado de la Autoridad que firma.

3.3.2.1.4 Algoritmo de Firma (*signature*)

Contiene el identificador del algoritmo que se usa para validar la firma del CAtr.

3.3.2.1.5 Número de Serie (*serialNumber*)

Los emisores deben asegurar que cada CAtr que emitan tenga un número de serie que sea distinto. Se deben utilizar números enteros positivos.

3.3.2.1.6 Período de validez (*attrCertValidityPeriod*)

Se deben completar dos campos que indican las fechas antes (*notBeforeTime*) y después (*notBeforeTime*) de las cuales el certificado no es válido.

3.3.2.1.7 Lista de atributos (*attributes*)

Los atributos proveen información acerca del portador. Cada atributo indicado cuenta con su identificador de tipo y los valores de cada uno. Existe una lista de tipos de atributos predefinidos que detallaremos más abajo.

3.3.2.1.8 Identificador único del emisor (*issuerUniqueID*)

La norma sugiere no utilizar este campo.

3.3.2.1.9 Extensiones (*extensions*)

Las extensiones proveen información sobre el CAtr en sí y son opcionales. Entre ellas figuran:

- Identidad de Auditoría (*auditIdentity*): se puede utilizar para que no sea posible identificar al portador de un CAtr.
- Objetivo (*acTargeting*): con esta extensión se puede indicar en qué servidores o servicios se puede presentar este CAtr.
- Identificador de la Clave del Emisor (*authorityKeyIdentifier*): se puede utilizar para ayudar al sistema verificador a validar la firma de la Autoridad que emitió el CAtr.
- Información de Acceso a la Autoridad (*authorityInfoAccess*): permite dar información sobre dónde se puede consultar el estado de revocación de los Certificados mediante protocolo OCSP.
- Puntos de Distribución de Listas de Certificados Revocados (*crlDistributionPoints*): otorga información al sistema verificador para que pueda consultar el estado de revocación de este CAtr.
- Revocación No Disponible (*noRevAvail*): se puede utilizar para indicar que no hay información sobre el estado de revocación de este CAtr.

3.3.3 Tipos de atributos

Para especificar los atributos (competencias) propiamente dichos, la RFC 5755 propone una serie de tipologías predefinidas, entre las cuales están:

- **Información para Servicios de Autenticación (*Service Authentication Information*):** este tipo de atributo identifica al poseedor ante un servidor/servicio con un *nombre* y

un *dato de autenticación relevante*. Normalmente se usa con un par *nombre de usuario/contraseña*.

- **Identidad de Acceso (*Access Identity*):** es similar al anterior sólo que no se incluye el *dato de autenticación relevante*.
- **Identidad con Cargo/Costo (*Charging Identity*):**se utiliza para indicarle al sistema verificador que se debe cobrar al poseedor del CAtr por el servicio que se le va a prestar.
- **Grupo (*Group*):** indica pertenencia a un grupo.
- **Rol (*Role*):**contiene información de asignación de roles al poseedor del CAtr. La sintaxis de uso de este atributo incluye un nombre del role y un nombre de la entidad con autoridad para otorgar dicho rol.
- **Autorización de seguridad (*Security Clearance*):** es el tipo de atributo más complejo y se usa para indicar el nivel de seguridad al que el portador está autorizado a acceder. Entre los datos que contiene se incluye un campo (cadena de bits) en donde se puede marcar: *unmarked* (0), *unclassified* (1), *restricted* (2), *confidential* (3), *secret* (4), *topSecret* (5).

3.3.4 Perfil del Certificado Digital del portador

El CCP debe tener marcado el bit *digitalSignature* en la extensión "uso de clave" (*keyUsage*) y para evitar confusiones con los números de serie y la revocaciones, se recomienda que la Autoridad que emite los CCP no sea la misma que emite los CAtr.

3.3.5 Validación de un Certificado de Atributos

- Para ser válido, un CAtr debe cumplir con lo siguiente:
- Cuando el portador use un CCP para autenticarse ante el verificador del CAtr toda la cadena de certificación debe ser validada.
- La firma del CAtr debe ser criptográficamente correcta, al igual que la del CCP y toda la cadena de confianza.
- El CCP debe cumplir con el perfil indicado anteriormente.
- El emisor del CAtr debe ser de confianza.
- La fecha y hora en que se evalúa el certificado debe estar comprendido entre las fechas "no antes" y "no después" del certificado.
- Si el CAtr contuviera una extensión crítica no soportada, se debe rechazar.

3.3.5.1 Verificaciones opcionales

El sistema verificador puede aplicar criterios adicionales específicos para rechazar un CAtr. Por ejemplo, si contuviera atributos para los cuáles la autoridad que los emitió no estuviera habilitada a hacerlo (ver: Controles AA).

3.3.6 Controles AA

En el momento de la validación del CAtr se puede comprobar si la autoridad que emitió el certificado está autorizada a otorgar los atributos que allí figuren. Para ello se puede utilizar la extensión "*AAControls*" en el certificado de la Autoridad de Certificación o la Autoridad de Competencia.

Con esta extensión, se puede especificar entre otras cosas:

- La lista de atributos que está autorizado a otorgar.
- La lista de atributos que no está autorizado a otorgar.
- Qué hacer si figuran atributos no especificados en ninguna de las listas anteriores.

La utilización de este control no es obligatoria y la Ley Argentina no hace mención al respecto.

3.3.7 Consideraciones de seguridad

Ante todo, la protección de la clave privada de la Autoridad de Certificación que emite el CAtr es lo que debe primar. Un atacante en poder de la misma puede enmascararse como ésta y emitir CAtr falsos o modificar el estado de revocación. Por lo cual, en caso de detectar una situación de compromiso, todos los CAtr deben ser revocados ya que, finalmente, la pérdida de la clave privada conllevaría a la imposibilidad de producir estados de revocación o realizar renovaciones.

Otro punto relevante tiene que ver con la precisión con la que se comparan los nombres con los que se vinculan los CCP con los CAtr. Un error puede derivar en la aceptación de un CAtr inválido o el rechazo de uno válido. La especificación X.500 define reglas para comparar nombres que no diferencian entre mayúsculas y minúsculas, codificación, set de caracteres o espacios en blanco al principio y final de las cadenas. Para este caso no son lo suficientemente estrictas, por tanto, se sugiere hacer una comparación bit a bit.

En el caso de que el CCP y el CAtr estuvieran vinculados por el campo *baseCertificateID*, es fundamental que la configuración de la PKI impida la existencia de dos AC con el mismo nombre ya que se podría montar un rogué AC que emita CAtrs que puedan aceptarse como válidos.

Por último, como se mencionó en el apartado anterior, el sistema que verifica un CAtr debe comprobar que el emisor del mismo esté autorizado a otorgar los atributos que en él figuren.

3.3.8 Revocación

Existen dos esquemas de revocaciones definidos, la Autoridad emisora puede elegir la que mejor se adapte al propósito de los CAtr que emite.

- Sin revocación: utilizando la extensión *nonRevAvail* se puede indicar que no se emite información de revocación para estos certificados. Todas las Autoridades deben implementar este esquema por más que sí emitan información de revocación. Esto se debe a que la norma apunta al uso de los CAtr como mecanismos de autorización de muy corta vida, por ejemplo, el tiempo de duración de una sesión de usuario.
- Con revocación: un CAtr puede apuntar a una fuente de información de estado de revocación utilizando la extensión *authorityInfoAccess* o *crlDistributionPoints.*

4 CAdES - CMS Advance Electronic Signatures

CMS (Cryptographic Message Sintax) es el estándar de la IETF para proteger mensajes de manera criptográfica y puede ser usado para cualquier forma de dato digital. Está basado en la sintaxis de PKCS#7 que deriva del estándar de protección de emails y su arquitectura está construida en base a la gestión de claves con certificados.

CAdES [11] fue introducido por la Unión Europea y extiende a CMS especificando varios perfiles adicionales para agregar capacidades importantes siendo, la más destacada, la validación a largo plazo para poder probar la validez de una firma mucho tiempo después de que fuera creada.

La filosofía de CAdES incluye el concepto de *políticas de firma* que se pueden usar para establecer los parámetros técnicos de validación de una firma electrónica. En este sentido, se pueden presentar dos escenarios. Uno, en el que se consigue un resultado consistente de la

validación ya que la política de firma que utiliza el verificador está explícitamente indicada por el firmante o es fácilmente deducible a partir del tipo de dato firmado u otro, en el que la política de validación no es explícita y conlleve a que los resultados de quienes validen la firma puedan ser disímiles. Por tanto, CAdES ofrece dos formas de firma: CAdES Explicit Policy-Based Electronic Signature (CAdES-EPES) y CAdES Basic Electronic Signature (CAdES-BES). En cualquiera de los casos es el firmante quien determina cuál forma usar.

4.1 CAdES-BES

Una firma con esta forma contiene:

- Los datos firmados (por ejemplo, un documento).
- Datos de validación: Certificados Digitales de Clave Pública (la cadena de confianza), el estado de revocación de cada uno de ellos y sellos de tiempo confiables aplicados a la firma.
- Una colección obligatoria de atributos firmados entre los cuales figuran el tipo de contenido (*Content-type*), resumen del mensaje del campo anterior (*Message-digest*) y el certificado digital con el que se firma.
- El valor de firma digital computada sobre los datos de usuario y los atributos firmados que estuvieran presentes.
- Opcionalmente puede contener una colección adicional de atributos firmados y no firmados, como por ejemplo, el campo para indicar si se trata de una contra-firma (*CounterSignature*) con la que se realizar una firma sobre otra, la ubicación geográfica del firmante a la hora de firmar (*Signer-Location*) o, los **atributos del firmante (Signer-Attributes)**, que se pueden especificar uno a uno como un listado o, sino, incluyendo un **Certificado de Atributos**.

CAdES-BES es el formato mínimo de firma electrónica y no provee información suficiente para verificar a largo plazo pero hasta ahora satisface las directivas legales de la Unión Europea.

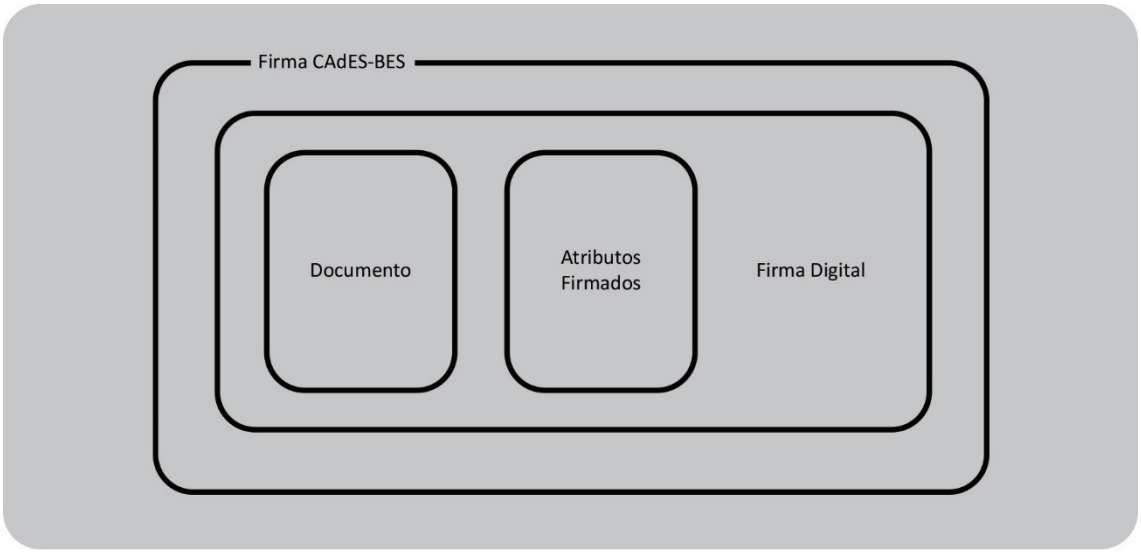

Ilustración 6 - CAdES-BES

4.2 CAdES-EPES

Este formato es igual al anterior sólo que agrega un atributo firmado en el que se indica la política de firma (*sigPolicyID*) con la que se debe validar la misma.

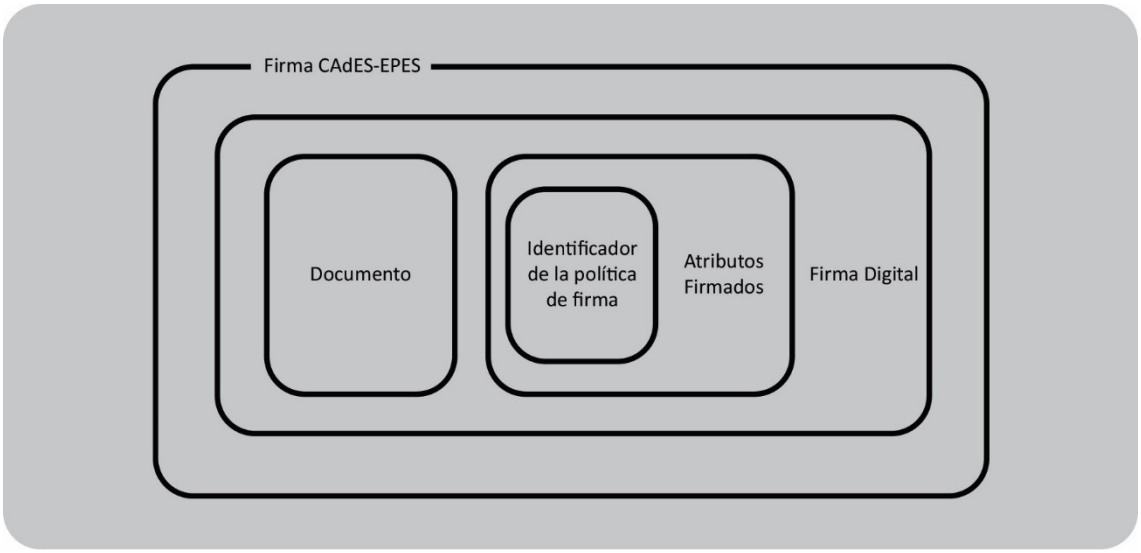

Ilustración 7 - CAdES-EPES

4.3 Perfiles CAdES

Existen distintos formatos que proveen información adicional a la firma. Cada uno agrega una propiedad más al anterior.

4.3.1 CAdES-T (con tiempo)

Incluye un atributo no firmado donde se especifica el momento de la firma. El mismo puede provenir de una fuente confiable como un servicio proveedor se sellos de tiempo, en cuyo caso se incluyen datos adicionales para poder validarlo como tal.

4.3.2 CAdES-C (Completo)

Agrega al perfil anterior la referencia completa a la cadena de confianza de los certificados digitales involucrados y las respuestas de las consultas del estado de revocación (OCSP) que se hizo para cada uno.

4.3.3 CAdES-X Long (Extendido)

Incorpora a la firma todos los certificados de la cadena de validación, las listas de revocación (CRLs) y las respuestas de las consultas del estado de revocación (OCSP) que se hizo para cada uno.

4.3.4 CAdES-A (Archivo)

Adiciona un atributo donde se indica el sello temporal del momento en que se archivó la firma.

5 Marco legal

La Firma Digital consiste en un mecanismo por el cual se pueden rubricar documentos electrónicos de manera que el autor de esos datos pueda ser identificado inequívocamente, permitiendo que en el ámbito jurídico posea la misma validez que la firma hológrafa, inclusive existiendo casos donde la firma hológrafa fue falsificada, mientras que la Firma Digital no tiene hasta la fecha antecedentes de repudio. En este sentido, la Firma Digital garantiza mayor seguridad.

5.1 Ley de Firma Digital

Ley Nº 25.506 [12] sancionada el 14 de Noviembre de 2001 legisló sobre la firma electrónica, la firma digital, el documento digital y su eficacia jurídica, incorporando estos conceptos al derecho argentino.

Dicha normativa define a la Firma Digital en su artículo segundo como el **"resultado de aplicar a un documento digital un procedimiento matemático que requiere información de exclusivo conocimiento del firmante, encontrándose ésta bajo su absoluto control. La Firma Digital debe ser susceptible de verificación por terceras partes, tal que dicha verificación simultáneamente permita identificar al firmante y detectar cualquier alteración del documento digital posterior a su firma".**

Conforme a la "Ley de Firma Digital", si un documento firmado digitalmente es verificado correctamente, se presume salvo prueba en contrario, que proviene del suscriptor del certificado asociado y que no fue modificado.

Además, incorpora la posibilidad de otorgar actos y contratos, con pleno valor jurídico, mediante documentos digitales y firmarlos digitalmente. Se equipara la validez del soporte electrónico a los documentos manuscritos tradicionales exigidos en forma escrita, con la Firma Digital como modo para suscribirlos.

El texto de la Ley, en su artículo tercero, indica que cuando se requiera una firma manuscrita, esa exigencia también quedará satisfecha por una Firma Digital. Este principio es aplicable a los casos en que la Ley establece la obligación de firmar o prescribe consecuencias para su ausencia.

Para la actual Ley de "Firma Digital" existen dos maneras de realizar este proceso:

1. Usando certificados emitidos por un Certificador Licenciado y suscribiendo a sus políticas (a este proceso se lo denomina Firma Digital);
2. La otra forma es realizar esta acción con certificados emitidos por un certificador no licenciado, suscribiendo a sus políticas, (Firma Electrónica).

5.2 Decisión Administrativa Nº 927 del 30 de octubre de 2014

Esta Decisión Administrativa de la Jefatura de Gabinete de Ministros [13], establece el marco normativo de Firma Digital aplicable al otorgamiento y revocación de las licencias a los certificadores que así lo soliciten, conforme a los requisitos y procedimientos establecidos en ella y sus Anexos. Define también la Infraestructura de Firma Digital de la República Argentina, estableciendo una Autoridad de Certificación Raíz y un sólo nivel de Autoridades de Certificación Subordinadas.

Por otro lado detalla nuevos servicios para la Firma Digital: los Sellos de Tiempo, para ofrecer un mecanismo de referencia temporal confiable y; los llamados Sellos de Competencia, con los que se busca extender el significado de una rúbrica para indicar el rol, cargo o atribución similar con el cual el firmante realiza el acto de firmar un documento electrónico. Para implementar este último mecanismo, propone el uso de Certificados de Atributos x509 y detalla, en el Anexo IV, el perfil que deben tener los mismos.

Para estos nuevos servicios, aparecen nuevas autoridades que extiendan los Sellos correspondientes, las cuales pueden constituirse tanto como Autoridades de Certificación o como Autoridades de Registro. De esta manera, el esquema de entes de confianza podría ser como se ve en la **Ilustración 8**.

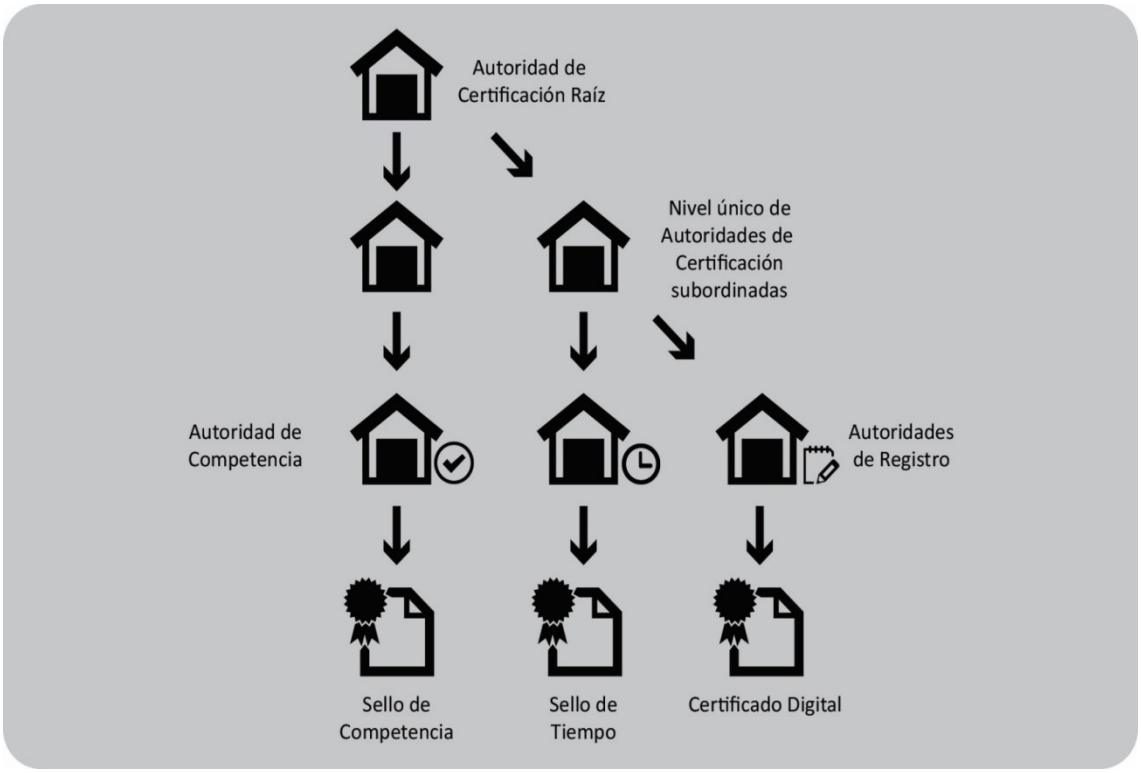

Ilustración 8 - Esquema de Autoridades

6 Estudio de alternativas para la emisión de Sellos de Competencia

El estándar propuesto por la RFC 5755 - *An Internet Attribute Certificate Profile for Authorization*, data de enero de 2010. Han pasado sólo cuatro años y son pocas las implementaciones que se han hecho en los distintos lenguajes de programación existentes. Concretamente, existen solamente dos y ambas basadas en Java(aunque una de ellas también tiene su versión C#). Se trata, en realidad, de dos "proveedores" de JCA y JCE que además cuentan con una extensa API de seguridad.

JCA (*Java Cryptography Architecture*) es parte de Java 2 desde JDK 1.1 y JCE (*Java Cryptography Extension*) agrega funcionalidades a la anterior a partir de JDK 1.4. En general, *Java Cryptography*se refiere a un conjunto de interfaces y clases que definen el motor de seguridad Java pero no lo implementan. Quienes sí lo hacen, en cambio, son los "proveedores".

6.1 Bouncy Castle Crypto API

BouncyCastle("BC") es una completa API de seguridad que es desarrollada y mantenida por "*The Legion of Bouncy Castle Inc.*", una organización australiana cuya figura legal tiene la forma de una asociación caritativa y que subsiste a partir de donaciones de los usuarios o de la venta de servicios de soporte. Esta suite de librerías fue creada en Java bajo la premisa del *open source* a finales de 1990 y recién en 2006 sacó su primera versión para C# y, más allá de lo cómico que pueda resultar traducir su nombre (Castillo Inflable), se trata de la única librería que contempla de manera absoluta la seguridad para Java (y la excede) y que a su vez es de libre acceso, utilización y distribución.

En detalle, BC es:

- Una API liviana de criptografía para Java y C#.
- Un proveedor para JCA y JCE.
- Una librería para leer y escribir objetos de tipo ASN.1 (*Abstract Syntax Notation One* o Notación Sintáctica Abstracta 1 es una norma para representar datos u objetos independientemente de la máquina que se esté usando y sus formas de representación internas. Es un protocolo de nivel de presentación en el modelo OSI. Los certificados X509 son estructuras de este tipo).
- Una API liviana para el protocolo de Seguridad de Capa de Transporte.
- Un generador y procesador de
    - Certificados Digitales X509 y archivos PKCS#12.
    - Certificados de Atributos X509.
    - CMS (*Cryptography Message Sintax*), el estándar de cifrado de datos digitales.
    - OSCP (*Online Certificate Status Protocol*).
    - TSP (*Timestamping*), para sellar una firma digital con la una estampa de tiempo confiable.
    - CMP (*Certificate Management Protocol*), un mecanismo para gestionar certificados X509.
    - OpenPGP.
- Toda la librería y el código es libre.

6.1.1 Licencia

Se otorga permiso, sin cargo, a cualquier persona que obtenga una copia de este software, documentos y archivos asociados, a llevar a cabo sin restricciones ni limitaciones de derechos: uso, copia, modificación, mezcla, distribución, sub-licenciamiento o venta, sujeto a la siguiente condición:

Tanto el texto de otorgamiento de permisos anterior como una nota que deslinda de responsabilidades y comunica que el software se entrega sin ninguna garantía deben incluirse en cualquier copia donde se haga uso del mismo.

6.2 IAIK Crypto Toolkit

El Instituto de Comunicaciones y Procesamiento de Información Aplicados (IAIK en alemán) de la Universidad de Tecnología de Graz, Austria, creó una suite de seguridad basada en Java que hoy cuenta con una extensa lista de herramientas, servicios y aplicaciones. Dentro del abanico de soluciones, se encuentran:

- Un proveedor JCA/JCE con gran variedad de servicios criptográficos y algoritmos.
- Está equipada con librería ASN.1.
- Ofrece soporte completo para PKCS.
- Permite la construcción íntegra de una PKI.
- Cuenta con implementaciones para curvas elípticas.
- Incluye un proveedor para dispositivos PKCS#11, haciendo posible la integración y comunicación de Java con *tokens* USB, HSM o cualquier módulo de hardware tipo *Cryptoki*.
- Tiene una solución para firma de documentos electrónicos mediante CAdES y PAdES.
- Además se ofrecen muchos otros productos de seguridad para dispositivos móviles, módulos IP, componentes RFID, etc.

6.2.1 Licencia

Este software se puede adquirir de manera gratuita sólo para su evaluación o para propósitos educativos o de investigación. Para todo uso comercial se debe comprar.

6.3 Emisión de un Sello de Competencia (Certificado de Atributos)

6.3.1 Perfil de un Sello de Competencia

Como se dijera anteriormente, un Sello de Competencia es un Certificado de Atributos X509 que cumple con el perfil propuesto en el Anexo IV de la DA 927 para integrar la Firma Digital Argentina. A continuación detallaremos los campos que deben utilizarse y los valores que deberían tener los mismos.

Campo	**Descripción**
Versión (*version*)	Versión del estándar del Certificado de Atributos. Debe contener el valor 2 (correspondiente a la tercera version)
Número de Serie (*serialNumber*)	Contiene un número asignado por la Autoridad de Competencia a cada certificado. Dicho número DEBE ser único para cada certificado emitido por dicha Autoridad.
Algoritmo de Firma (*Signature*)	DEBE contener el identificador de objeto (OID) del algoritmo y, si fueran necesarios, los parámetros asociados usados por la

	Autoridad de Competencia para firmar el certificado. Este identificador DEBE ser alguno de los definidos en el [RFC4055] para RSA, [RFC5480] para curvas elípticas o [RFC5758] para DSA y ECDSA.
Nombre Distintivo del Emisor (*Issuer*)	DEBE identificar a la organización responsable de la emisión del sello de competencia. El contenido de este campo DEBE coincidir con el indicado en el campo del "distinguishedName" correspondiente al "subject" del certificado de la Autoridad de Competencia que fuera emitido por la Autoridad Certificante del certificador licenciado.
Validez (Desde, Hasta) (*attrCertValidityPeriod* (*notBefore*, *notAfter*))	El período de la validez es el intervalo de tiempo durante el cual el sello de competencia se considera válido. Este intervalo dependerá de la validez del atributo correspondiente. El campo se representa como una secuencia de dos fechas: "notBefore": fecha en que el período de validez del certificado comienza. "notAfter": fecha en que el período de validez del certificado termina. El período de validez de un certificado es el período de tiempo desde "notBefore" hasta "notAfter" inclusive. Una autoridad de competencia NO DEBE emitir un certificado con vencimiento posterior al de su propio certificado.
Nombre Distintivo del Titular (*holder*)	El campo "subject" identifica la entidad asociada a la competencia contenida en el certificado. DEBE contener el nombre distintivo del titular. Dicho nombre DEBE ser único para cada titular de certificado emitido por una autoridad de competencia durante todo el tiempo de vida del mismo. La identidad del suscriptor DEBE quedar especificada utilizando los siguientes atributos: Nombre común (OID 2.5.4.3: commonName): nombre de la Persona Jurídica Pública o Privada o de la Persona Física tal como figura en su document de identidad. Número de serie (OID 2.5.4.5: serialNumber): se debe indicar el tipo de identificación tributaria o laboral "CUIT/CUIL" y su número. **Estas indicaciones hacen suponer que se debe usar la opción *entityName* mencionada en el punto 3.3.2.1.2 pero, teniendo**

	en cuenta el perfil de un CCP de la misma DA927, faltaría agregar el código del país.
Competencia (*attributes*)	Este campo contiene las competencias asociadas al titular, que se está certificando. Pueden tratarse de competencias profesionales, relaciones laborales o similar. **La DA no es precisa en cuanto a qué tipo de atributo de los que figura en la RFC 5755 es el que se debe usar para indicar las competencias. Asumiremos por eltexto del cuerpo de la DA 927 (en su párrafo 13) que el tipo "rol" (*role*) mencionado en el el punto 3.3.3de este trabajo es la forma más conveniente de hacerlo.**

6.3.2 Requisitos para emitir un Sello de Competencia

Para poder emitir un Sello de Competencia, es necesario configurar un entorno que simule una situación real. Básicamente, lo que se requiere es contar previamente con tres Certificados Digitales adicionales:

- Uno que corresponda a la Autoridad de Certificación. Éste va a ser auto-firmado para hacer las veces también de una autoridad raíz y simplificar el ejemplo.
- Otro para representar a la Autoridad de Competencia que estará firmado por la Autoridad de Certificación.
- Finalmente, uno que sea el certificado patronímico de una Persona Física a la cual se le asignará una *competencia*. Este certificado es el que estará relacionado con el Sello de Competencia a través del campo "*holder"* de este último y represente a su titular.

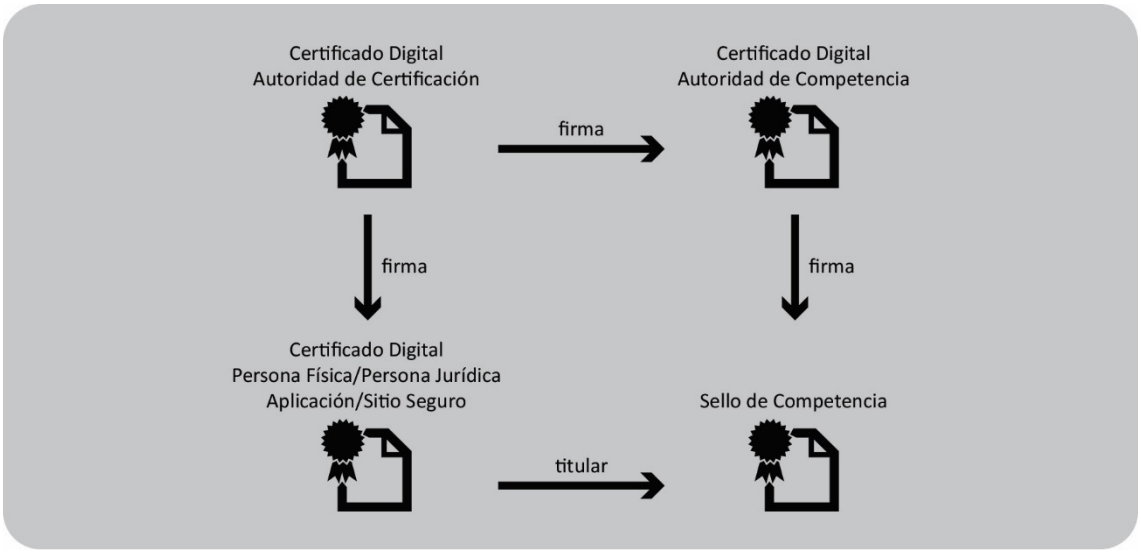

Ilustración 9 - Esquema de certificación

A fin de cumplir con este esquema, haremos uso de la herramienta *openSSL* para generar los tres certificados adicionales. En este sentido, primero se configurarán los perfiles de los certificados en un archivo de texto que luego se usará como entrada a los comandos de

openSSL. Se puede consultar el contenido de dichos archivos en el anexo I. Los pasos a seguir son:

1. Creación de un Certificado Digital auto-firmado para la **Autoridad de Certificación**

```
openssl req -x509 -config crearCertificadoAC.cnf -newkey rsa:4096 -sha1 -out AutoridadCertificacion.crt -outform PEM
```

2. Creación de una Solicitud de Firma de Certificado para la **Autoridad de Competencia**

```
openssl req -config crearCertificadoACOMP.cnf -newkey rsa:2048 -sha1 -out AutoridadCompetencia.csr -keyout AutoridadCompetenciaKey.pem -outform PEM
```

3. Firma del Certificado para la **Autoridad de Competencia**

```
openssl ca -config firmarCertificado.cnf -policy signing_policy -extensions signing_req -out AutoridadCompetencia.pem -infiles AutoridadCompetencia.csr
```

4. Convertir un certificado y una clave en formato PEM a PKCS#12

```
openssl pkcs12 -name AutoridadCompetencia -export -out AutoridadCompetencia.p12 -inkey AutoridadCompetenciaKey.pem -in AutoridadCompetencia.pem
```

5. Crear un Java Key Store para usar un certificado y su clave privada desde un programa Java

```
keytool -importkeystore -destkeystore AutoridadCertificacion.jks -srckeystore AutoridadCertificacion.p12 -srcstoretype pkcs12 -alias AutoridadCertificacion
```

6. Creación de una Solicitud de Firma de Certificado para la **Persona Física**

```
openssl req -config crearCertificadoPF.cnf -newkey rsa:2048 -sha1 -out PersonaFisica.csr -keyout PersonaFisicaKey.pem -outform PEM
```

7. Firma del Certificado para la **Persona Física**

```
openssl ca -config firmarCertificadoPF.cnf -policy signing_policy -extensions signing_req -out PersonaFisica.pem -infiles PersonaFisica.csr
```

8. Convertir el certificado deformato PEM a CER

```
openssl x509 -inform PEM -in PersonaFisica.pem -outform DER -out PersonaFisica.cer
```

Finalizados estos pasos contamos con los siguientes archivos, necesarios para poder emitir un Sello de Competencia desde un programa Java:

AutoridadCompetencia.jk **Java Key Store:** contiene el certificado y la clave

s	privada de la Autoridad de Competencia con los cuales se firmará el Sello de Competencia.
PersonaFisica.cer	**Certificado Digital:** certificado patronímico de una Persona Física a la cuál se le otorgará un Sello de Competencia que estará realacionado con este certificado.

6.3.3 Programa para emitir un Sello de Competencia con BouncyCastle

```
package ar.com.bcid.competencia;


import...


publicclass SelloCompetenciaConBC {


privatestaticfinal String BC = BouncyCastleProvider.PROVIDER_NAME;


publicstaticvoid main(String[] args){


        Security.addProvider(new BouncyCastleProvider());


        SelloCompetenciaConBC sc =new SelloCompetenciaConBC();

        sc.test();


}


privatevoid test(){


try{

// Certificado de la Persona Física

            CertificateFactory factory = CertificateFactory.getInstance("X.509");

            X509Certificate certificadoPersonaFisica =

(X509Certificate) factory.generateCertificate(

new FileInputStream("C:/Certificados/PersonaFisica.cer")

);


// Certificado y Clave de la Autoridad de Competencia

            String ksPassAutoridadCompetencia ="ninguna";


            KeyStore ksAutoridadCompetencia = KeyStore.getInstance("JKS");
```

```
            ksAutoridadCompetencia.load(
new FileInputStream(
new File("C:/Certificados/AutoridadCompetencia.jks")
),
                    ksPassAutoridadCompetencia.toCharArray()
);


            X509Certificate certificadoAutoridadCompetencia =
(X509Certificate) ksAutoridadCompetencia
.getCertificate("AutoridadCompetencia");

            String alPassAutoridadCompetencia ="ninguna";

            KeyPair kpAutoridadCompetencia =
                getPrivateKey(ksAutoridadCompetencia,
"AutoridadCompetencia",
                                alPassAutoridadCompetencia.toCharArray()
);

            PrivateKey privateKeyAutoridadCompetencia =
                kpAutoridadCompetencia.getPrivate();

// Creación del Certificado de Atributos
            AttributeCertificateHolder holder =
new AttributeCertificateHolder(
new X500Name(certificadoPersonaFisica.getSubjectDN().getName())
);

            X509v2AttributeCertificateBuilder generador =
new X509v2AttributeCertificateBuilder(
                    holder,
new AttributeCertificateIssuer(
new X500Name(
```

```
                            certificadoAutoridadCompetencia.getSubjectDN().getName()
)
),
                    BigInteger.ONE,
new Date(System.currentTimeMillis()),
new Date(System.currentTimeMillis()
+(1000l*60l*60l*24l*356l))// Le damos validez por 1 año
);

// Asignamos los atributos propiamente dichos mediante la estructura RoleSyntax
            GeneralName roleName =
new GeneralName(
                    GeneralName.uniformResourceIdentifier,
"Especialista en Seguridad Informática"
);
            GeneralNames authName =
new GeneralNames(
new GeneralName(
                        GeneralName.rfc822Name,"Instituto Universitario Aeronáutico")
);
            RoleSyntax roleSyntax =new RoleSyntax(authName, roleName);
            generador.addAttribute(new ASN1ObjectIdentifier("2.5.4.72"), roleSyntax);



            ContentSigner sigGen =
new JcaContentSignerBuilder("SHA1WithRSAEncryption")
.setProvider(BC)
.build(privateKeyAutoridadCompetencia);

// Firmamos el Certificado de Atributos
            X509AttributeCertificateHolder selloCompetencia = generador.build(sigGen);

// Mostramos por consola el Sello de Competencia codificado en Base64
```

```
            System.out.println(
new String(Base64.getEncoder().encode(selloCompetencia.getEncoded()))
            );


}catch(FileNotFoundException e){
            e.printStackTrace();
}catch(CertificateException e){
            e.printStackTrace();
}catch(KeyStoreException e){
            e.printStackTrace();
}catch(NoSuchAlgorithmException e){
            e.printStackTrace();
}catch(IOException e){
            e.printStackTrace();
}catch(OperatorCreationException e){
            e.printStackTrace();
}catch(CertException e){
            e.printStackTrace();
}
}


public KeyPair getPrivateKey(KeyStore keystore, String alias,
char[] password){


try{
            Key key = keystore.getKey(alias, password);
if(key instanceof PrivateKey){
                X509Certificate cert =
(X509Certificate) keystore.getCertificate(alias);
                PublicKey publicKey = cert.getPublicKey();
returnnew KeyPair(publicKey,(PrivateKey) key);
}
}catch(UnrecoverableKeyException e){
```

```
}catch(NoSuchAlgorithmException e){

}catch(KeyStoreException e){

}

returnnull;

}

}
```

Tabla 1 - Emitir un Sello de Competencia con BouncyCastle

```
Version: 2

Holder: {entityName: directoryName: C=AR,CN=Gonzalo Arguello,serialNumber=CUIL20317420669}

Issuer: {issuerName: directoryName: C=AR,O=Instituto Universitario Aeronautico,CN=Autoridad de
Competencia - IUA,serialNumber=CUIT309876543210}

Signature algorithm: sha1WithRSAEncryption (1.2.840.113549.1.1.5)

Serial number: 2323971292

Valid not before time: Sat May 30 16:09:22 ART 2015

Valid not after  time: Sun May 29 16:09:22 ART 2016

Attributes: 1


Certificate Fingerprint (MD5)  : 9E:EB:A2:65:03:D7:51:F6:1E:0A:04:7C:BF:10:B8:DF

Certificate Fingerprint (SHA-1): ED:16:D5:A1:A7:6D:3A:1E:E5:27:A0:2A:5A:49:ED:88:0C:BB:11:26


Rol: Especialista en Seguridad Informática

Emitido por: Instituto Universitario Aeronáutico
```

Tabla 2 - Información del Sello de Competencia

6.3.4 Programa para emitir un Sello de Competencia con IAIK

```
package ar.com.bcid.competencia;


import...


publicclass SelloCompetenciaConIiak {


privatestaticfinal String BC = BouncyCastleProvider.PROVIDER_NAME;


publicstaticvoid main(String[] args){


        SelloCompetenciaConIaik sc =new SelloCompetenciaConIaik();

        sc.test();


}


privatevoid test(){
```

```
try{
// Certificado de la Persona Física
            CertificateFactory factory = CertificateFactory.getInstance("X.509");
            X509Certificate certificadoPersonaFisica =
(X509Certificate) factory.generateCertificate(
new FileInputStream("C:/Certificados/PersonaFisica.cer")
);

// Certificado y Clave de la Autoridad de Competencia
            String ksPassAutoridadCompetencia ="ninguna";

            KeyStore ksAutoridadCompetencia = KeyStore.getInstance("JKS");

            ksAutoridadCompetencia.load(
new FileInputStream(
new File("C:/Certificados/AutoridadCompetencia.jks")
),
                    ksPassAutoridadCompetencia.toCharArray()
);

            X509Certificate certificadoAutoridadCompetencia =
(X509Certificate) ksAutoridadCompetencia
.getCertificate("AutoridadCompetencia");

            String alPassAutoridadCompetencia ="ninguna";

            KeyPair kpAutoridadCompetencia =
                getPrivateKey(ksAutoridadCompetencia,
"AutoridadCompetencia",
                                alPassAutoridadCompetencia.toCharArray()
);

            PrivateKey privateKeyAutoridadCompetencia =
```

```
                kpAutoridadCompetencia.getPrivate();


// Creación del Certificado de Atributos


            Holder holder =new Holder();
            Name holderDN =
new Name(
                            certificadoPersonaFisica.getSubjectX500Principal().
                                getEncoded()
);
            GeneralName holderName =
new GeneralName(GeneralName.directoryName,holderDN);
            GeneralNames holderNames =
new GeneralNames(holderName);
            holder.setEntityName(holderNames);


            Name issuerDN =
new Name(
                            certificadoAutoridadCompetencia
.getSubjectDN().getName()
);
            GeneralName issuerName =
new GeneralName(GeneralName.directoryName,issuerDN);
            GeneralNames issuerNames =
new GeneralNames(issuerName);
            AttCertIssuer issuer =
new V2Form(issuerNames);


            AttributeCertificate selloCompetencia =new AttributeCertificate();
selloCompetencia.setHolder(holder);
selloCompetencia.setIssuer(issuer);
selloCompetencia.setNotBeforeTime(new Date());
selloCompetencia.setNotAfterTime(new Date(System.currentTimeMillis()
```

```
+(1000*60*60*24*356)));// Le damos validez por 1 año
selloCompetencia.setSerialNumber(new BigInteger(32,new Random()));
selloCompetencia.setSignatureAlgorithm(AlgorithmID.sha1WithRSAEncryption);

            GeneralName roleName =
new GeneralName(
                            GeneralName.uniformResourceIdentifier,
"Especialista en Seguridad Informática"
);
            GeneralNames authName =
new GeneralNames(
new GeneralName(
                                    GeneralName.rfc822Name
,"Instituto Universitario Aeronáutico"
)
);

// Asignamos los Atributes con la estructura Role
            Role role =new Role(roleName);
            role.setRoleAuthority(authName);
selloCompetencia.addAttribute(new Attribute(role));

// Firmamos el Certificado de Atributos
selloCompetencia.sign(
                    AlgorithmID.sha1WithRSAEncryption,
                    privateKeyAutoridadCompetencia
);
// Mostramos por consola el Sello de Competencia codificado en Base64
            System.out.println(
new String(Base64.getEncoder().encode(selloCompetencia.getEncoded()))
);

}catch(FileNotFoundException e){
```

```
            e.printStackTrace();
}catch(CertificateException e){
            e.printStackTrace();
}catch(KeyStoreException e){
            e.printStackTrace();
}catch(NoSuchAlgorithmException e){
            e.printStackTrace();
}catch(IOException e){
            e.printStackTrace();
}catch(OperatorCreationException e){
            e.printStackTrace();
}catch(CertException e){
            e.printStackTrace();
}
}

public KeyPair getPrivateKey(KeyStore keystore, String alias,
char[] password){

try{
            Key key = keystore.getKey(alias, password);
if(key instanceof PrivateKey){
                X509Certificate cert =
(X509Certificate) keystore.getCertificate(alias);
                PublicKey publicKey = cert.getPublicKey();
returnnew KeyPair(publicKey,(PrivateKey) key);
}
}catch(UnrecoverableKeyException e){
}catch(NoSuchAlgorithmException e){
}catch(KeyStoreException e){
}
returnnull;
}
```

```
}
```

Tabla 3 - Emitir un Sello de Competencia con IAIK

```
Version: 2

Holder: {entityName: directoryName: serialNumber=CUIL20317420669,CN=Gonzalo Arguello,C=AR}

Issuer: {issuerName: directoryName: serialNumber=CUIT309876543210,CN=Autoridad de Competencia
- IUA,O=Instituto Universitario Aeronautico,C=AR}

Signature algorithm: sha1WithRSAEncryption (1.2.840.113549.1.1.5)

Serial number: 2649488829

Valid not before time: Sat May 30 16:03:39 ART 2015

Valid not after  time: Sun May 29 16:03:39 ART 2016

Attributes: 1



Certificate Fingerprint (SHA-1): E3:A1:C6:8A:03:78:AC:58:5B:3E:BB:1F:75:48:C4:77:F5:71:A3:B9



Rol: Especialista en Seguridad Informática

Emitido por: Instituto Universitario Aeronáutico
```

Tabla 4 - Información del Sello de Competencia

6.4 Elección de una alternativa

Como se pudo apreciar, desde el punto de vista de la programación es muy sencillo generar un Certificado de Atributos con cualquiera de las librerías utilizadas ya que son pocas las líneas de código que se necesitan para lograr el cometido. Más aún, al ser las dos librerías implementaciones de un mismo estándar con estructuras genéricas, las clases y los métodos utilizan denominaciones muy similares, aunque IAIK ofrece algunos mecanismos algo más intuitivos, lo que reduce la complejidad a la hora de usarlo.

Sin embargo, hay dos cosas que harían decantar nuestra decisión de utilizar BouncyCastle en un presunto proyecto. En primer lugar, respeta la libertad de todos los programadores/usuarios que adquirieron el producto y, por tanto, una vez obtenido el mismo, puede ser usado, copiado, estudiado, modificado y redistribuido libremente de varias formas. De esta manera podemos encarar nuevos proyectos tanto de investigación como comerciales con total seguridad de que estamos en pleno derecho de uso del producto.

En segundo lugar, y muy ligado al punto anterior, la disponibilidad de documentación *on-line* es mayor para *BouncyCastle* que para *IAIK*. Esto es un gran punto a favor en un momento como este en el que nos encontramos en una etapa exploratoria respecto de cómo implementar lo que propone el nuevo marco legal de la Firma Digital de la República Argentina.

En el cuadro siguiente se muestran los distintos puntos de comparación considerados en este estudio de alternativas.

Producto	Licencia	Complejidad de uso	Disponibilidad de documentación	Soporte por parte del desarrollador	Soporte de la comunidad

IAIK Crypto Toolkit	Libre (para propósitos educativos o de investigación) Paga (para proyectos comerciales)	Medio	Baja	Pago	Bajo
BouncyCastle	**Libre (para todo uso)**	**Alta**	**Alta**	**Pago**	**Alto**

7 Firma Digital de un documento electrónico con un Sello de Competencia

Para firmar digitalmente un documento, hoy por hoy, se maneja un conjunto de estándares de firma avanzada (AdES: Advanced Electronic Signature) aceptados globalmente y que cubren un espectro casi universal de propósitos. Estos son: CAdES, PAdES o XAdES. El primero es en cierto modo el más genérico de los tres (por eso ampliamos su explicación en el capítulo 4) y es el que utilizaremos en el presente trabajo a modo de demostración.

Por su lado, PAdES es el estándar para firma de archivos PDF. Entre sus diversas formas y configuraciones cuenta también con compatibilidad CAdES, es decir que se puede aplicar una firma con estas mismas especificaciones incluyendo las extensiones -T, -C, -X, -X-L, -A y –LT.

Finalmente, XAdES sigue el mismo camino que los anteriores para unificar el universo de firmas digitales en los archivos XML. Es decir, amplía el estándar XML-DSig especificando perfiles precisos para adecuarlos a la firma avanzada (AdES).

Antes de proceder al ejemplo, recordemos que un documento se firma con la clave privada asociada a un Certificado Digital específico, sin embargo, los Certificados de Atributos no tiene ninguna clave privada asociada, por lo tanto, ¿Cómo se firma un documento con un certificado de este tipo? Como se explicó en el punto 4.1, entre las extensiones opcionales del estándar CAdES (punto 5.11.3 de RFC 5126) se pueden incluir roles del firmante de dos formas: listando los roles uno a uno o; **adjuntando en la firma un Certificado de Atributos x509 (nuestro Sello de Competencia).** Pues bien, en definitiva, la forma de hacerlo es utilizando el Certificado Digital del sujeto (titular) para firmar el documento (porque cuenta con una clave privada asociada) y adjuntar a la firma CAdES el Certificado de Atributos tal como lo indica el estándar.

7.1 Software para Firma Digital

La demostración la haremos utilizando el *Cliente @Firma del Gobierno de España*. Se trata de una aplicación Java que permite realizar todo tipo de firmas digitales con los estándares más conocidos y que, a su vez, se ofrece bajo la licencia GPL v2 por lo cual aprovechamos esta condición para modificarlo de tal manera que podamos extender nuestras firmas con los Sellos de Competencia.

A continuación mostraremos las clases modificadas y el código desarrollado para lograr el propósito del presente trabajo.

En primer lugar, identificamos el método en el cual se especifican los atributos firmados como lo requiere CAdES o, dicho de otra forma, donde se genera la información que acompaña a la firma. El método se denomina *generateSignerInfo()* y la clase en donde se encuentra el mismo

es *CAdESUtils.java*. Pero antes de modificar ésta, debemos "hacer llegar" a ese punto el Certificado de Atributos que queremos adjuntar. Investigando el código, encontramos que a esta función se le pasa un parámetro de tipo *CAdESSignerMetadata*. Esta clase es portadora de la información sobre el firmante e implementa el punto 5.11.2 del estándar CAdES por lo cual constituye el lugar ideal para incorporar también lo estipulado en el punto 5.11.3: los atributos del firmante.

Modificamos entonces esta última clase para que se pueda instanciar un objeto de tipo *CAdESSignerMetadata* que también contenga un Certificado de Atributos. El mismo llegará al constructor como una cadena de caracteres codificada en Base64. Decidimos hacerlo de esta forma porque, como veremos más adelante, la configuración general de la firma se hace a partir de una lista de propiedades de tipo *<clave-valor>,* por lo cual se podría obtener el Certificado de Atributos de un repositorio, codificarlo en Base64 y enviarlo al software que realizará la firma dentro de dicha lista. Veamos en las secciones sombreadas del código que aparece a continuación cómo queda la modificación propuesta.

```
package es.gob.afirma.signers.cades;


import java.io.IOException;

import java.util.List;


import org.bouncycastle.cert.X509AttributeCertificateHolder;

import org.bouncycastle.util.encoders.Base64;


publicfinalclass CAdESSignerMetadata {


privatefinal CAdESSignerLocation signerLocation;

privatefinal CAdESSignerAttribute signerAttribute;


privatestaticfinalint POSTAL_ADDRESS_MAX_LINES =6;




public CAdESSignerMetadata(final String country,

final String locality,

final List<String> address,

final String base64AttributeCertificate)throws IOException

{

this.signerLocation =new CAdESSignerLocation(country, locality, address);

this.signerAttribute =new CAdESSignerAttribute(base64AttributeCertificate);
```

```
}

public CAdESSignerLocation getSignerLocation(){
returnthis.signerLocation;
}

public CAdESSignerAttribute getSignerAttribute(){
returnthis.signerAttribute;
}

//fragmento de código omitido

publicstaticfinalclass CAdESSignerAttribute{
private X509AttributeCertificateHolder attributeCertificate;

    CAdESSignerAttribute (final String base64AttributeCertificate)
throws IOException
{
try{
if(!base64AttributeCertificate.trim().equalsIgnoreCase("")&&
                base64AttributeCertificate !=null){
this.attributeCertificate =
new X509AttributeCertificateHolder(
                    Base64.decode(base64AttributeCertificate)
);
}
}catch(IOException e){
thrownew IOException("Error generando el Certificado de
                        Atributos codificado en base64");
}

}

public X509AttributeCertificateHolder getAttributeCertificate(){
```

```
return attributeCertificate;

}

}

}
```

Tabla 5 - Modificación de la clase CAdESSignerMetadata

Esta nueva versión de *CAdESSignerMetadata* ahora es capaz de leer una cadena de caracteres codificada en Base64 y construir a partir de allí un objeto de tipo *X509AttributeCertificateHolder*, estructura de *BouncyCastle* que hace de envoltorio para los Certificado de Atributos X509.

El paso siguiente consiste en adaptar una clase auxiliar que transforma objetos Java en estructuras de tipo ASN.1. Como mencionamos anteriormente, la Notación Sintáctica Abstracta 1 es una norma para representar datos u objetos independientemente de la plataforma y la tecnología. Tanto los certificados digitales como la firma CAdES son "estructuras de estructuras" ASN.1. Lo que buscamos entonces, es incluir en esta clase auxiliar un método que transforme el objeto de tipo *X509AttributeCertificateHolder* en uno con formato ASN.1 que podamos incluir en la firma CAdES.

Afortunadamente, *BouncyCastle* cuenta con dicho mecanismo a través de la clase *SignerAttribute* que implementa exactamente el punto 5.11.3 del estándar CAdES.

```
publicfinalclass CAdESSignerMetadataHelper {


private CAdESSignerMetadataHelper(){
// No instanciable
}


publicstatic CAdESSignerMetadata getCAdESSignerMetadata(
final Properties extraParams)throws IOException
{
if(extraParams ==null){
returnnull;
}
// signerLocationPostalAddress
final String postalAddressOneLine =
            extraParams.getProperty("signatureProductionPostalCode");
final String[] postalAddress = postalAddressOneLine ==null?
null:
                postalAddressOneLine.split("\n");
```

```
// signerLocationCountryName
final String country = extraParams.getProperty("signatureProductionCountry");

// signerLocationLocalityName
final String locality = extraParams.getProperty("signatureProductionCity");

// signerAttribute
final String base64AttributeCertificate =
extraParams.getProperty("signerAttribute");

if(postalAddress !=null|| country !=null|| locality !=null
|| base64AttributeCertificate !=null){
returnnew CAdESSignerMetadata(
                country,
                locality,
                postalAddress !=null?
                    Arrays.asList(postalAddress):
null,
                base64AttributeCertificate
);
}
returnnull;
}

// fragmento de código omitido

publicstatic SignerAttribute getSignerAttribute(
final CAdESSignerMetadata.CAdESSignerAttribute csa
){
if(csa ==null)
returnnull;
if(csa.getAttributeCertificate()==null)
returnnull;
```

```
returnnew SignerAttribute(csa.getAttributeCertificate().toASN1Structure());
}
}
```

Tabla 6 - Modificación de la clase auxiliar CAdESSignerMetadataHelper

Como se puede ver, esta clase es la que recibe el listado de propiedades mencionado, toma aquellas que pertenecen a los datos del firmante, construye un objeto de tipo *CAdESSignerMetada* y ofrece dos métodos para obtener estos datos en formato ASN.1. Se han resaltado fragmentos relevantes del código para una mejor comprensión de las modificaciones realizadas.

Bien, hasta ahora hemos logrado incluir en el modelo del *Cliente @Firma* los datos relacionados con los atributos del firmante. Resta solamente hacer que el software incorpore esto en la firma CAdES. La forma de hacerlo es modificando el método *generateSignerInfo()* de la clase *CAdESUtils*, el cual se encarga de elaborar un vector de objetos ASN.1 con todos los atributos firmados obligatorios y opcionales que define el estándar CAdES (en realidad no todos ya que los denominados *SignerAttributes* los estamos incluyendo en este trabajo de investigación).

```
publicstatic ASN1EncodableVector generateSignerInfo(
final Certificate cert,
final String digestAlgorithmName,
finalbyte[] data,
final AdESPolicy policy,
finalboolean signingCertificateV2,
finalbyte[] dataDigest,
final Date signDate,
finalboolean padesMode,
final String contentType,
final String contentDescription,
final List<CommitmentTypeIndicationBean> ctis,
final CAdESSignerMetadata csm,
finalboolean isCountersign)
throws NoSuchAlgorithmException,
        IOException, CertificateEncodingException {

// fragmento de código omitido
```

```
// id-aa-ets-signerAttr

if(csm !=null&&

       CAdESSignerMetadataHelper.

        getSignerAttribute(csm.getSignerAttribute())!=null)

{

        contexExpecific.add(

new Attribute(

                PKCSObjectIdentifiers.id_aa_ets_signerAttr,

new DERSet(CAdESSignerMetadataHelper.

                    getSignerAttribute(

                        csm.getSignerAttribute()

)

)

)

);

}
```

Tabla 7 - Modificación de la clase CAdESUtils

De esta sencilla forma hemos logrado adaptar un software de firma digital de código abierto para que contemple también el uso de los Sellos de Competencia propuestos por la DA 927.

7.1.1 Demostración de firma digital

El programa a continuación utiliza una clase de prueba del Cliente @Firma que hemos modificado para firmar con el Certificado Digital de la Persona Física que creamos en el punto 6.3.2 y extender la rúbrica con el Sello de Competencia emitido con BouncyCastle en el ejemplo 6.3.3.

El objetivo es firmar con CAdES un documento simple llamado *prueba.txt*.

```
package es.gob.afirma.test.cades;


import...


publicfinalclass TestCAdESconSelloCompetencia {


privatestaticfinal String CERT_PATH ="PersonaFisica.p12";

privatestaticfinal String CERT_PASS ="ninguna";

privatestaticfinal String CERT_ALIAS ="PersonaFisica";
```

```
privatestaticfinal String[] DATA_FILES ={
"prueba.txt"
};

privatestaticfinal List<byte[]> DATA =new ArrayList<byte[]>(2);
static{
for(final String dataFile : DATA_FILES){
try{
                DATA.add(AOUtil.getDataFromInputStream(
                    TestCAdESconSelloCompetencia.class.getResourceAsStream(dataFile))
);
}catch(final IOException e){
                Logger.getLogger("es.gob.afirma")
.severe("No se ha podido cargar el fichero de pruebas: "+ dataFile);
                DATA.add(null);
}
}
}

privatestaticfinal Properties[] CADES_MODES;

static{
final Properties p1 =new Properties();
        p1.setProperty("format", AOSignConstants.SIGN_FORMAT_CADES);
        p1.setProperty("mode", AOSignConstants.SIGN_MODE_IMPLICIT);
        p1.setProperty("signerAttribute",
"MIIChjCCAW4CAQEwSKFGpEQwQjEYMBYGA1UEBRMPQ1VJTDIwMzE3NDIwNjY5MRkwFwYDVQQDDBBHb"
+"256YWxvIEFyZ3VlbGxvMQswCQYDVQQGEwJBUqCBhzCBhKSBgTB/MRkwFwYDVQQFExBDVUlUMzA5OD"
+"c2NTQzMjEwMScwJQYDVQQDDB5BdXRvcmlkYWQgZGUgQ29tcGV0ZW5jaWEgLSBJVUExLDAqBgNVBAo"
+"MI0luc3RpdHV0byBVbml2ZXJzaXRhcmlvIEFlcm9uYXV0aWNvMQswCQYDVQQGEwJBUjANBgkqhkiG"
+"9w0BAQUFAAIFALLBxjEwIhgPMjAxNTA1MzAxODE2NTVaGA8yMDE1MDYwNzE4NTcyNFowWzBZBgNVB"
+"EgxUjBQoCWBI0luc3RpdHV0byBVbml2ZXJzaXRhcmlvIEFlcm9u4XV0aWNvoSeGJUVzcGVjaWFsaX"
+"N0YSBlbiBTZWd1cmlkYWQgSW5mb3Jt4XRpY2EwDQYJKoZIhvcNAQEFBQADggEBAFLLtFBks9K4bXF"
+"mnL2Tbi7pwQAASX7Wh4ilO6vaFJfw8fQtxwzpgWDLNcZt0WglId5yRVVY/O5lTm12xTCmLiVY75bP"
+"uOeMfvrvpmIU+dX4kHT1mgb2JUjDsXiPUyCmRBFUtsCnvuD/UFaAi87p5OraN/zTQKwggvTl+R/SW"
```

```
+"mQoS0nvZLeIYvLXZhUcDK7srW87iPXWYxunP8qL5cs8sYwvJI9Kp2UtgmhP6Z1ZTwDF5RxvTp7IEF"
+"XEQpncrg2zcirAjNKOxiXAjsYv9alQuvc10q5LejCjhLaMzWEg0iegkxLGapR7TjXQ6Gc9fBmCO/R"
+"Ws2784UreZLP9caJSkpg=");

        CADES_MODES =new Properties[]{
                p1
};
}

/** Algoritmos de firma a probar. */
privatefinalstatic String[] ALGOS =new String[]{
        AOSignConstants.SIGN_ALGORITHM_SHA1WITHRSA,
};

/**
     * Prueba de firma convencional.
     * @throws Exception en cualquier error
     */
    @SuppressWarnings("static-method")
    @Test
publicvoid testSignature(){
try{
        Logger.getLogger("es.gob.afirma").setLevel(Level.WARNING);
final PrivateKeyEntry pke;
final X509Certificate cert;

final KeyStore ks = KeyStore.getInstance("PKCS12");
        ks.load(ClassLoader.getSystemResourceAsStream(CERT_PATH), CERT_PASS.toCharArray());
        pke =(PrivateKeyEntry) ks.getEntry(
                                CERT_ALIAS,
new KeyStore.PasswordProtection(CERT_PASS.toCharArray())
);
        cert =(X509Certificate) ks.getCertificate(CERT_ALIAS);

final AOSigner signer =new AOCAdESSigner();
```

```
        String prueba;
for(final Properties extraParams : CADES_MODES){
for(final String algo : ALGOS){
for(int i =0; i < DATA_FILES.length; i++){
if(DATA.get(i)==null){
continue;
}

                    prueba ="Firma CAdES del fichero "+ DATA_FILES[i]+" en modo '"+
                            extraParams.getProperty("mode")+
"' con el algoritmo ': "+
                            algo +
"' y politica '"+
                            extraParams.getProperty("policyIdentifier")+
"'";

                    System.out.println(prueba);

finalbyte[] result = signer.sign(
                                                        DATA.get(i),
                                                        algo,
                                                        pke.getPrivateKey(),
                                                        pke.getCertificateChain(),
                                                        extraParams
);

final File saveFile = File.createTempFile(algo +"-",".csig");
final OutputStream os =new FileOutputStream(saveFile);
                    os.write(result);
                    os.flush();
                    os.close();
                    System.out.println(
"Temporal para comprobacion manual: "+ saveFile.getAbsolutePath()
);

                    Assert.assertNotNull(prueba, result);
```

```
                    Assert.assertTrue(signer.isSign(result));
                    Assert.assertTrue(
                        CAdESValidator.isCAdESValid(
                            result,
                            AOSignConstants.CMS_CONTENTTYPE_SIGNEDDATA,
true
)
);


                    AOTreeModel tree = signer.getSignersStructure(result,false);
                    Assert.assertEquals("Datos",
((AOTreeNode) tree.getRoot()).getUserObject());
                    Assert.assertEquals(
"Gonzalo Arguello",
((AOTreeNode) tree.getRoot()).getChildAt(0).getUserObject()
);


                    tree = signer.getSignersStructure(result,true);
                    Assert.assertEquals("Datos",
((AOTreeNode) tree.getRoot()).getUserObject());
final AOSimpleSignInfo simpleSignInfo =
(AOSimpleSignInfo)((AOTreeNode) tree.getRoot())
.getChildAt(0).getUserObject();


                    Assert.assertNotNull(simpleSignInfo.getSigningTime());
                    Assert.assertEquals(cert, simpleSignInfo.getCerts()[0]);
                    System.out.println(prueba +": OK");
}
}
}


        signer.sign(
                DATA.get(0),
"SHA1withRSA",
                pke.getPrivateKey(),
                pke.getCertificateChain(),
```

```
null

);


}catch(Exception e){

            e.printStackTrace();

}


}


}
```

Tabla 8 - Firma CAdES del documento "prueba.txt"

Como mencionamos en el punto anterior, para firmar con este software la configuración general se hace a partir de una lista de propiedades de tipo *<clave-valor>* en la que decidimos colocar el *Sello de Competencia* codificado en Base64 ya que estos parámetros pueden ser configurados del lado del servidor y luego enviados al Cliente @Firma. En el ejemplo de arriba, esta configuración está inserta en el objeto de tipo *Properties* de la sección sombreada.

La firma resultante se almacena en un archivo *.csig* que podemos verificar con el siguiente código.

```
package ar.com.bcid.firma;


import...


publicclass VerInfoFirma {


privatestaticfinal String BC = BouncyCastleProvider.PROVIDER_NAME;


publicstaticvoid main(String[] args){

        Security.addProvider(new BouncyCastleProvider());


        VerInfoFirma f =new VerInfoFirma();

        f.ver();

}


privatevoid ver(){

try{

            File f =new File(

"C:/Users/Gonzalo/AppData/Local/Temp/”+
```

```
”SHA1withRSA-55619334594153738.csig");
byte[] signedBytes =newbyte[(int) f.length()];
            DataInputStream in =new DataInputStream(new FileInputStream(f));
            in.readFully(signedBytes);
            in.close();

            CMSSignedData s =new CMSSignedData(signedBytes);
            Store certStore = s.getCertificates();
            SignerInformationStore signers = s.getSignerInfos();

            Collection c = signers.getSigners();
            Iterator it = c.iterator();

int verified =0;
while(it.hasNext()){
                SignerInformation signer =(SignerInformation) it.next();
                Collection certCollection = certStore.getMatches(signer.getSID());

                Iterator certIt = certCollection.iterator();
                X509CertificateHolder certificadoPersonaFisica =
(X509CertificateHolder) certIt.next();

// Extraemos el atributo CAdES "SignerAttribute" en donde se encuentra el Sello de Competencia
// Utilizamos su OID para obtenerlo
                ASN1Sequence seq = ASN1Sequence.getInstance(
                        signer.getSignedAttributes()
.get(PKCSObjectIdentifiers.id_aa_ets_signerAttr)
.getAttrValues().getObjectAt(0));
                ASN1TaggedObject tag = ASN1TaggedObject.getInstance(seq.getObjectAt(0));

// Obtenemos el Sello de Competencia y generamos una instancia del mismo
                X509AttributeCertificateHolder selloCompetencia =
new X509AttributeCertificateHolder(tag.getObject().getEncoded());

// Verificamos el Certificado de la Persona Física (Titular)
if(signer.verify(new JcaSimpleSignerInfoVerifierBuilder()
```

```
.setProvider(BC).build(certificadoPersonaFisica))){
                    System.out.println("CERTIFICADO VERIFICADO");
}

// Mostramos por consola los certificados extraídos de la firma
// Usamos la librería de IAIK sólo porque tiene una implementación de toString() más clara

                X509Certificate cpf =
new X509Certificate(certificadoPersonaFisica.getEncoded());
                System.out.println("########################");
                System.out.println("CERTIFICADO DEL TITULAR");
                System.out.println(cpf);

                AttributeCertificate ac =
new AttributeCertificate(tag.getObject().getEncoded());
                Role rol =
new Role(ac.getAttribute(ObjectID.role)
.getAttributeValue().toASN1Object());
                System.out.println("####################");
                System.out.println("SELLO DE COMPETENCIA");
                System.out.println(ac);
                System.out.println("Rol: "+ rol.getRoleName().getName());
                System.out.println("Emitido por: "
+((GeneralName)rol.getRoleAuthority()
.getNames().nextElement()).getName());

}

}catch(Exception e){
            e.printStackTrace();
}
}
}
```

Tabla 9 - Verificación del contenido de la firma

Finalmente podemos demostrar el resultado esperado del presente trabajo con la salida por consola del programa anterior. **Como se puede ver a continuación, es posible extraer de la firma digital tanto el Certificado del titular como el Sello de Competencia. De esta manera se podría corroborar, en una situación real, las atribuciones con las cuales una persona firmó determinado documento.**

```
CERTIFICADO VERIFICADO



########################

CERTIFICADO DEL TITULAR

Version: 3

Serial number: 1

Signature algorithm: sha1WithRSAEncryption (1.2.840.113549.1.1.5)

Issuer: serialNumber=CUIT309876543210,CN=AC - IUA Especialidad en Seguridad
Informatica,O=Instituto Universitarioa Aeronautico,C=AR

Valid not before: Tue May 26 16:56:54 ART 2015

      not after: Fri May 23 16:56:54 ART 2025

Subject: serialNumber=CUIL20317420669,CN=Gonzalo Arguello,C=AR

Sun RSA public key, 2048 bits

                                                                                modulus:
2063002050323102355007388531732407155514421728812737240905948520437470672651553210626228260834310029399257152772820508516095739398480622007347512588407009688340075475589960155829851159354207864981724741093379911154214147062623620094598152352611641990759409284878537362109092885460417630317028999527281987744270429839875339380966301518667943343726750881559913817810537597242055712839500366772018508211689451818066170995593066979048254526747873288678120855318233271442665385823385725585342326249492389438355221972034288598724802220054330568448406145160398025855601541297610164586177280788180464678591008922873565119128 9

  public exponent: 65537

Certificate Fingerprint (MD5)  : C8:BA:C8:F4:00:3E:82:61:4D:69:16:36:B9:E1:A3:A5

Certificate Fingerprint (SHA-1): FB:8C:25:86:6B:ED:A9:A2:91:3B:B0:E4:CE:12:BC:CF:53:1F:7F:30



Extensions: 4



####################

SELLO DE COMPETENCIA

Version: 2

Holder: {entityName: directoryName: serialNumber=CUIL20317420669,CN=Gonzalo Arguello,C=AR}

Issuer: {issuerName: directoryName: serialNumber=CUIT309876543210,CN=Autoridad de Competencia -
IUA,O=Instituto Universitario Aeronautico,C=AR}

Signature algorithm: sha1WithRSAEncryption (1.2.840.113549.1.1.5)

Serial number: 3342037517

Valid not before time: Sat May 30 15:14:10 ART 2015
```

```
Valid not after  time: Sun Jun 07 15:54:39 ART 2015

Attributes: 1


Certificate Fingerprint (MD5)  : 5A:E8:21:75:68:4D:18:6E:A3:57:94:66:93:77:4A:5C

Certificate Fingerprint (SHA-1): 52:BF:DF:E7:FB:03:C9:0B:1F:14:22:7A:95:CD:80:2B:93:E7:A3:30


Rol: Especialista en Seguridad Informática

Emitido por: Instituto Universitario Aeronáutico
```

Tabla 10 - Salida por consola

8 Conclusiones

8.1 Conclusión general

Con este proyecto se buscaba realizar una implementación concreta y efectiva de un Sello de Competencia adecuado a la normativa más reciente de la Infraestructura de Firma Digital de la República Argentina y explorar las posibilidades de utilizarlo para extender la firma digital de un documento electrónico con los roles o atribuciones otorgados al firmante como lo propone la Ley.

El proceso de investigación comenzó con el estudio en profundidad de una Infraestructura de Clave Pública (PKI) y continuó con el examen exhaustivo de los estándares tecnológicos que definen a los Certificados de Atributos (RFC 5755) y a la firma electrónica avanzada CAdES (RFC 5126). La compresión cabal de estos conceptos permitió iniciar el estudio técnico de las alternativas existentes para su implementación. De esta forma, encontramos en la librería *BouncyCastle* el conjunto ideal de herramientas para emitir, de manera sencilla, un Certificado de Atributos con el perfil de un Sello de Competencia como lo indica la DA 927 y que cumple con la premisa del presente trabajo de utilizar exclusivamente software *open-source*.

Logrado el objetivo anterior, se debía demostrar la factibilidad de firmar digitalmente un documento electrónico con el Sello de Competencia emitido y, para ello, se hizo uso del software Cliente @Firma. Esta herramienta provee la funcionalidad de firmar con CAdES pero no incluye los mecanismos necesarios para hacerlo utilizando un Certificado de Atributos en conjunto con un Certificado Digital. Esa modificación fue realizada en el marco del presente trabajo examinando en detalle la implementación de las distintas clases involucradas y añadiendo el código necesario para incorporar esta nueva función cuya ejecución de prueba arrojó un resultado exitoso.

Finalmente, para comprobar la correcta implementación de todo lo anterior, se pudo extraer la información de la firma y corroborar que de la misma se puede recuperar el Sello de Competencia tal como fue emitido.

De esta forma, se logró cumplir con cada uno de los objetivos específicos del proyecto y alcanzar el resultado esperado. Queda así sentada la base de lo que sería una primera propuesta para abordar la implementación conjunta de la Firma Digital de la República Argentina y los nuevos servicios que plantea.

8.2 Trabajos futuros

Conociendo la mecánica para emitir un Sello de Competencia, un paso a seguir, sería el desarrollo de una arquitectura que aborde la solución de gestión de los mismos. En ese sentido sería necesaria una infraestructura que implemente una Autoridad de Competencia que administre las solicitudes de Sellos por parte de los usuarios y gobierne su ciclo de vida desde la puesta en vigencia hasta su invalidación o revocación, incluyendo los servicios de consulta de estado de los mismos.

Un punto importante a considerar es que esta Autoridad de Competencia podría ser el repositorio central de los certificados de donde sus titulares podrían obtenerlos al momento de firmar. Esto es posible ya que los mismos no están asociados a una clave privada y por lo tanto, no requeriría su custodia en el equipo o dispositivo del usuario.

9 Referencias bibliográficas

[01] León, Jeffrey, apunte “RSA”, asignatura “Codes and Cryptography”, DMS, University of Illinois at Chicago, 2008

http://homepages.math.uic.edu/~leon/mcs425-s08/handouts/RSA.pdf

Fecha de consulta: Mayo 2015

[02] Agustín, artículo “Entender RSA”, Foro Técnico, Kriptópolis - Criptografía y Seguridad, 2012

http://www.kriptopolis.com/entender-rsa

Fecha de consulta: Mayo 2015

[03] International Telecommunication Union, “ITU-TX.509 Information technology - Open systems interconnection - The Directory: Public-key and attribute certificate frameworks”, Telecomm Standardization Sector of ITU, 2012

http://www.itu.int/rec/T-REC-X.509-201210-I/es

Fecha de consulta: Mayo 2015

[04] The Internet Engineering Task Force (IETF®) - OpenPGP Message Format - RFC4880, 2007

http://www.ietf.org/rfc/rfc4880.txt

Fecha de consulta: Mayo 2015

[05] The Internet Engineering Task Force (IETF®) - Internet X.509 Public Key Infrastructure - Certificate and Certificate Revocation List (CRL) Profile - RFC3280, 2002

http://www.ietf.org/rfc/rfc3280.txt

Fecha de consulta: Mayo 2015

[06] International Telecommunication Union, “Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)”, Telecomm Standardization Sector of ITU, 2009

http://www.itu.int/rec/T-REC-X.690-200811-I/es

Fecha de consulta: Mayo 2015

[07] Sriram Ranganathan, Sans Institute, “Key and Certificate Management in Public Key Infrastructure Technology”, 2001

http://www.sans.org/reading-room/whitepapers/vpns/key-certificate-management-public-key-infrastructure-technology-735

Fecha de consulta: Mayo 2015

[08] The Internet Engineering Task Force (IETF®) - A TCP/IP Tutorial, 1991

http://tools.ietf.org/rfc/rfc1180.txt

Fecha de consulta: Mayo 2015

[09] International Telecommunication Union, “X.500: Tecnología de la información - Interconexión de sistemas abiertos - El directorio: Visión de conjunto de conceptos, modelos y servicios”, Telecomm Standardization Sector of ITU, 2012

http://www.itu.int/rec/T-REC-X.500-201210-I/es

Fecha de consulta: Mayo 2015

[10] The Internet Engineering Task Force (IETF®) - An Internet Attribute Certificate Profile for Authorization – RFC5755, 2010

http://tools.ietf.org/pdf/rfc5755.pdf

Fecha de consulta: Abril 2015

[11] The Internet Engineering Task Force (IETF®) - Network Working Group - CMS Advanced Electronic Signatures (CAdES) – RFC5126, 2008

https://tools.ietf.org/pdf/rfc5126.pdf

Fecha de consulta: Abril 2015

[12] Congreso de la Nación Argentina, Ley Nacional, Ley de Firma Digital, 2001

http://infoleg.mecon.gov.ar/infolegInternet/anexos/70000-74999/70749/norma.htm

Fecha de consulta: Abril 2015

[13] Presidencia de la Nación Argentina, Jefatura de Gabinete de Ministros, Decisión Administrativa 927/2014 (y anexos), 2014

http://www.infoleg.gov.ar/infolegInternet/anexos/235000-239999/237642/norma.htm

Fecha de consulta: Abril 2015

[14] Fernando Boiero, Carlos Ignacio Tapia "Aplicación de Firma Electrónica a procesos del IUA", 2014.

Fecha de consulta: Abril 2015

[15] Cliente @firma, Forja del Centro de Transferencia de Tecnología, Gobierno de España, 2015.

http://forja-ctt.administracionelectronica.gob.es/web/clienteafirma

Fecha de consulta: Abril 2015

10 Anexo I

Archivos de configuración OpenSSL

```
HOME            = .

RANDFILE        = $ENV::HOME/.rnd


####################################################################

[ ca ]

default_ca  = CA_default        # The default ca section


[ CA_default ]


default_days    = 3650          # how long to certify for

default_crl_days= 30            # how long before next CRL

default_md  = sha256            # use public key default MD

preserve    = no                # keep passed DN ordering


x509_extensions = ca_extensions     # The extensions to add to the cert


email_in_dn = no            # Don't concat the email in the DN

copy_extensions = copy          # Required to copy SANs from CSR to cert


####################################################################

[ req ]

default_bits        = 4096

default_keyfile     = AutoridadCertificacionKey.pem

distinguished_name  = ca_distinguished_name

x509_extensions     = ca_extensions

string_mask         = utf8only


####################################################################

[ ca_distinguished_name ]


countryName                             = País (código de 2 caracteres)
```

```
countryName_default             = AR


organizationName                = Nombre de la Organización

organizationName_default        = Instituto Universitarioa Aeronautico


commonName                              = Nombre Común

commonName_default              = AC - IUA Especialidad en Seguridad Informatica


serialNumber                                    = CUIT+Num Cuit

serialNumber_default                    = CUIT309876543210


####################################################################

[ ca_extensions ]


basicConstraints       = critical, CA:true

keyUsage               = keyCertSign, cRLSign

subjectKeyIdentifier   = hash

authorityKeyIdentifier = keyid:always, issuer
```

Archivo 1 - Configuración de perfil del certificado de la Autoridad de Certificación

```
HOME            = .

RANDFILE        = $ENV::HOME/.rnd


####################################################################

[ req ]

default_bits        = 2048

distinguished_name  = acomp_distinguished_name

req_extensions      = acomp_req_extensions

string_mask         = utf8only

version                         = 2


####################################################################

[ acomp_distinguished_name ]
```

```
countryName                     = País (código de 2 caracteres)
countryName_default     = AR

organizationName                = Nombre de la Organización
organizationName_default    = Instituto Universitario Aeronautico

commonName                      = Nombre del Servicio o Unidad Operativa
commonName_default      = Autoridad de Competencia - IUA

serialNumber                            = CUIT/CUIL+Num
serialNumber_default            = CUIT309876543210

#####################################################################
[ acomp_req_extensions ]

basicConstraints        = CA:FALSE
subjectKeyIdentifier    = hash
keyUsage                = nonRepudiation, keyCertSign, cRLSign
```

Archivo 2 - Configuración del perfil del certificado de la Autoridad de Competencia

```
HOME            = .
RANDFILE        = $ENV::HOME/.rnd

#####################################################################
[ req ]
default_bits        = 2048
distinguished_name  = pf_distinguished_name
req_extensions      = pf_req_extensions
string_mask         = utf8only
version                         = 2

#####################################################################
[ pf_distinguished_name ]
countryName                     = País (código de 2 caracteres)
```

```
countryName_default     = AR


commonName                      = Nombre/s y Apellidos/s (según documento de identidad)

commonName_default      =


serialNumber                            = CUIT/CUIT+Num

serialNumber_default            =


#################################################################

[ pf_req_extensions ]


basicConstraints        = CA:FALSE

subjectKeyIdentifier    = hash

keyUsage                =     nonRepudiation     ,     digitalSignature,     keyEncipherment,
dataEncipherment, keyAgreement, encipherOnly, decipherOnly
```

Archivo 3 - Configuración del perfil del certificado de la Persona Física

```
HOME            = .

RANDFILE        = $ENV::HOME/.rnd


#################################################################

[ ca ]

default_ca  = CA_default        # The default ca section


[ CA_default ]


default_days    = 3650          # how long to certify for

default_crl_days= 30            # how long before next CRL

default_md  = sha1        # use public key default MD

preserve    = no            # keep passed DN ordering


x509_extensions = ca_extensions     # The extensions to add to the cert


email_in_dn = no            # Don't concat the email in the DN

copy_extensions = copy          # Required to copy SANs from CSR to cert
```

```
base_dir    = .
certificate = $base_dir/AutoridadCertificacion.crt  # The CA certifcate
private_key = $base_dir/AutoridadCertificacionKey.pem   # The CA private key
new_certs_dir   = $base_dir     # Location for new certs after signing
database    = $base_dir/index.txt   # Database index file
serial      = $base_dir/serial.txt  # The current serial number

unique_subject  = yes            # Set to 'no' to allow creation of
                # several certificates with same subject.

####################################################################
[ req ]
default_bits        = 4096
default_keyfile     = AutoridadCertificacionKey.pem
distinguished_name  = ca_distinguished_name
x509_extensions     = ca_extensions
string_mask         = utf8only

####################################################################
[ ca_distinguished_name ]

countryName                         = País (código de 2 caracteres)
countryName_default             = AR

organizationName                = Nombre de la Organización
organizationName_default        = Instituto Universitarioa Aeronautico

commonName                          = Nombre Común
commonName_default              = AC - IUA Especialidad en Seguridad Informatica

serialNumber                                = CUIT+Num Cuit
serialNumber_default                    = CUIT309876543210
```

```
####################################################################

[ ca_extensions ]


basicConstraints = critical, CA:true

keyUsage = keyCertSign, cRLSign

subjectKeyIdentifier=hash

authorityKeyIdentifier=keyid:always, issuer


####################################################################

[ signing_policy ]

countryName             = supplied

organizationName        = supplied

commonName              = supplied

serialNumber            = supplied


####################################################################

[ signing_req ]

subjectKeyIdentifier=hash

authorityKeyIdentifier=keyid,issuer


basicConstraints = CA:FALSE

keyUsage = nonRepudiation , digitalSignature, keyEncipherment, dataEncipherment, keyAgreement,
encipherOnly, decipherOnly
```

Archivo 4 - Configuración de la firma del Certificado de la Autoridad de Competencia

```
HOME            = .

RANDFILE        = $ENV::HOME/.rnd


####################################################################

[ ca ]

default_ca  = CA_default        # The default ca section


[ CA_default ]
```

```
default_days    = 3650          # how long to certify for
default_crl_days= 30            # how long before next CRL
default_md  = sha1        # use public key default MD
preserve    = no            # keep passed DN ordering

x509_extensions = ca_extensions     # The extensions to add to the cert

email_in_dn = no            # Don't concat the email in the DN
copy_extensions = copy          # Required to copy SANs from CSR to cert

base_dir    = .
certificate = $base_dir/AutoridadCertificacion.crt  # The CA certifcate
private_key = $base_dir/AutoridadCertificacionKey.pem   # The CA private key
new_certs_dir   = $base_dir     # Location for new certs after signing
database    = $base_dir/index.txt   # Database index file
serial      = $base_dir/serial.txt  # The current serial number

unique_subject  = yes            # Set to 'no' to allow creation of
                # several certificates with same subject.

####################################################################
[ req ]
default_bits        = 4096
default_keyfile     = AutoridadCertificacionKey.pem
distinguished_name  = ca_distinguished_name
x509_extensions     = ca_extensions
string_mask         = utf8only

####################################################################
[ ca_distinguished_name ]

countryName                         = País (código de 2 caracteres)
```

```
countryName_default             = AR


organizationName                = Nombre de la Organización

organizationName_default        = Instituto Universitarioa Aeronautico


commonName                              = Nombre Común

commonName_default              = AC - IUA Especialidad en Seguridad Informatica


serialNumber                                    = CUIT+Num Cuit

serialNumber_default                    = CUIT309876543210




#####################################################################

[ ca_extensions ]


basicConstraints = critical, CA:true

keyUsage = keyCertSign, cRLSign

subjectKeyIdentifier=hash

authorityKeyIdentifier=keyid:always, issuer


#####################################################################

[ signing_policy ]

countryName             = supplied

commonName              = supplied

serialNumber            = supplied


#####################################################################

[ signing_req ]

subjectKeyIdentifier=hash

authorityKeyIdentifier=keyid,issuer


basicConstraints = CA:FALSE

keyUsage = nonRepudiation , digitalSignature, keyEncipherment, dataEncipherment, keyAgreement,
encipherOnly, decipherOnly
```

Archivo 5 - Configuración de la firma del Certificado de la Persona Física