Unión Internacional de Telecomunicaciones

UIT-T X.1080.0

(03/2017)

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

SERIE X: REDES DE DATOS, COMUNICACIONES DE SISTEMAS ABIERTOS Y SEGURIDAD

Seguridad de la información y de las redes – Telebiometría

Control de acceso para la protección de datos telebiométricos

Recomendación UIT-T X.1080.0

Unión Internacional de Telecomunicaciones

RECOMENDACIONES UIT-T DE LA SERIE X

REDES DE DATOS, COMUNICACIONES DE SISTEMAS ABIERTOS Y SEGURIDAD

REDES PÚBLICAS DE DATOS	X.1–X.199
INTERCONEXIÓN DE SISTEMAS ABIERTOS	X.200–X.299
INTERFUNCIONAMIENTO ENTRE REDES	X.300–X.399
SISTEMAS DE TRATAMIENTO DE MENSAJES	X.400–X.499
DIRECTORIO	X.500–X.599
GESTIÓN DE REDES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS Y ASPECTOS DE SISTEMAS	X.600–X.699
GESTIÓN DE INTERCONEXIÓN DE SISTEMAS ABIERTOS	X.700–X.799
SEGURIDAD	X.800–X.849
APLICACIONES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS	X.850–X.899
PROCESAMIENTO DISTRIBUIDO ABIERTO	X.900–X.999
SEGURIDAD DE LA INFORMACIÓN Y DE LAS REDES	
Aspectos generales de la seguridad	X.1000–X.1029
Seguridad de las redes	X.1030–X.1049
Gestión de la seguridad	X.1050–X.1069
Telebiometría	**X.1080–X.1099**
APLICACIONES Y SERVICIOS CON SEGURIDAD	
Seguridad en la multidifusión	X.1100–X.1109
Seguridad en la red residencial	X.1110–X.1119
Seguridad en las redes móviles	X.1120–X.1139
Seguridad en la web	X.1140–X.1149
Protocolos de seguridad	X.1150–X.1159
Seguridad en las comunicaciones punto a punto	X.1160–X.1169
Seguridad de la identidad en las redes	X.1170–X.1179
Seguridad en la TVIP	X.1180–X.1199
SEGURIDAD EN EL CIBERESPACIO	
Ciberseguridad	X.1200–X.1229
Lucha contra el correo basura	X.1230–X.1249
Gestión de identidades	X.1250–X.1279
APLICACIONES Y SERVICIOS CON SEGURIDAD	
Comunicaciones de emergencia	X.1300–X.1309
Seguridad en las redes de sensores ubicuos	X.1310–X.1339
INTERCAMBIO DE INFORMACIÓN DE CIBERSEGURIDAD	
Aspectos generales de la ciberseguridad	X.1500–X.1519
Intercambio de estados/vulnerabilidad	X.1520–X.1539
Intercambio de eventos/incidentes/heurística	X.1540–X.1549
Intercambio de políticas	X.1550–X.1559
Petición de heurística e información	X.1560–X.1569
Identificación y descubrimiento	X.1570–X.1579
Intercambio asegurado	X.1580–X.1589
SEGURIDAD DE LA COMPUTACIÓN EN NUBE	
Visión general de la seguridad de la computación en nube	X.1600–X.1601
Diseño de la seguridad de la computación en nube	X.1602–X.1639
Prácticas óptimas y directrices en materia de seguridad de la computación en nube	X.1640–X.1659
Aplicación práctica de la seguridad de la computación en nube	X.1660–X.1679
Otras cuestiones de seguridad de la computación en nube	X.1680–X.1699

Para más información, véase la Lista de Recomendaciones del UIT-T.

Recomendación UIT-T X.1080.0

Control de acceso para la protección de datos telebiométricos

Resumen

En la Recomendación UIT-T X.1080.0 se especifica cómo proteger la información telebiométrica contra el acceso no autorizado adoptando un enfoque orientado al servicio, donde sólo se facilita la información necesaria para un fin concreto, es decir, que el acceso no sólo se concede en función del *derecho a saber*, sino también en función de la *necesidad de conocer*.

La piedra angular de esta Recomendación es una especificación de atributo, incluida en un certificado de atributo o certificado de clave pública, que especifica detalladamente qué privilegios posee una entidad concreta para uno o más tipos de servicios.

La seguridad se da con un perfil de sintaxis de mensaje criptográfico (CMS), que garantiza la autentificación, la integridad y, cuando procede, la confidencialidad (encriptación).

Este perfil está previsto para dar seguridad a las especificaciones telebiométricas en general. El perfil supone la correcta implantación de una infraestructura de clave pública (PKI) y, al mismo tiempo, depende de ella.

Esta Recomendación también depende de la existencia de una infraestructura de gestión de privilegios (PMI).

Historia

Edición	Recomendación	Aprobación	Comisión de Estudio	ID único*
1.0	ITU-T X.1080.0	2017-03-30	17	11.1002/1000/13193

Palabras clave

Control de acceso, Diffie-Hellman, PKI, telebiometría.

* Para acceder a la Recomendación, sírvase digitar el URL http://handle.itu.int/ en el campo de dirección del navegador, seguido por el identificador único de la Recomendación. Por ejemplo, http://handle.itu.int/11.1002/1000/11830-en.

PREFACIO

La Unión Internacional de Telecomunicaciones (UIT) es el organismo especializado de las Naciones Unidas en el campo de las telecomunicaciones y de las tecnologías de la información y la comunicación. El Sector de Normalización de las Telecomunicaciones de la UIT (UIT-T) es un órgano permanente de la UIT. Este órgano estudia los aspectos técnicos, de explotación y tarifarios y publica Recomendaciones sobre los mismos, con miras a la normalización de las telecomunicaciones en el plano mundial.

La Asamblea Mundial de Normalización de las Telecomunicaciones (AMNT), que se celebra cada cuatro años, establece los temas que han de estudiar las Comisiones de Estudio del UIT-T, que a su vez producen Recomendaciones sobre dichos temas.

La aprobación de Recomendaciones por los Miembros del UIT-T es el objeto del procedimiento establecido en la Resolución 1 de la AMNT.

En ciertos sectores de la tecnología de la información que corresponden a la esfera de competencia del UIT-T, se preparan las normas necesarias en colaboración con la ISO y la CEI.

NOTA

En esta Recomendación, la expresión "Administración" se utiliza para designar, en forma abreviada, tanto una administración de telecomunicaciones como una empresa de explotación reconocida de telecomunicaciones.

La observancia de esta Recomendación es voluntaria. Ahora bien, la Recomendación puede contener ciertas disposiciones obligatorias (para asegurar, por ejemplo, la aplicabilidad o la interoperabilidad), por lo que la observancia se consigue con el cumplimiento exacto y puntual de todas las disposiciones obligatorias. La obligatoriedad de un elemento preceptivo o requisito se expresa mediante las frases "tener que, haber de, hay que + infinitivo" o el verbo principal en tiempo futuro simple de mandato, en modo afirmativo o negativo. El hecho de que se utilice esta formulación no entraña que la observancia se imponga a ninguna de las partes.

PROPIEDAD INTELECTUAL

La UIT señala a la atención la posibilidad de que la utilización o aplicación de la presente Recomendación suponga el empleo de un derecho de propiedad intelectual reivindicado. La UIT no adopta ninguna posición en cuanto a la demostración, validez o aplicabilidad de los derechos de propiedad intelectual reivindicados, ya sea por los miembros de la UIT o por terceros ajenos al proceso de elaboración de Recomendaciones.

En la fecha de aprobación de la presente Recomendación, la UIT no ha recibido notificación de propiedad intelectual, protegida por patente, que puede ser necesaria para aplicar esta Recomendación. Sin embargo, debe señalarse a los usuarios que puede que esta información no se encuentre totalmente actualizada al respecto, por lo que se les insta encarecidamente a consultar la base de datos sobre patentes de la TSB en la dirección http://www.itu.int/ITU-T/ipr/.

ÍNDICE

Página

Introducción

Cuando se obtienen datos telebiométricos de las personas, se corre el riesgo de infringir la privacidad.

Puede haber varios motivos para proteger esa información, que puede dar acceso a una empresa u organización o puede ser de naturaleza sensible, que limita su distribución.

La protección contra la divulgación no deseada de datos telebiométricos tiene dos dimensiones principales:

– la protección de los datos durante la transmisión, generalmente mediante encriptación, y la protección de los datos almacenados; y

– el control de acceso a los datos almacenados.

Si bien los sistemas telebiométricos deben tener un nivel alto de seguridad en cuanto a confidencialidad (encriptación), autentificación, integridad, protección física, utilización de cortafuegos, programas de protección contra virus, etc., también es necesario crear un sistema complejo de control de acceso a la información almacenada, en particular la información de carácter personal. Este último aspecto es particularmente importante en el caso de los sistemas telebiométricos.

Los esquemas de control de acceso generales suelen pecar de considerar principalmente el *derecho* (o su ausencia) a la información, pero no contemplan la *necesidad de saber*. Por necesidad de saber se entiende que no basta con tener derecho a ver los datos, sino que se ha de demostrar que esa información se va a utilizar con fines legítimos.

La información sólo se debe facilitar para el fin a que está destinada. La información médica de un paciente se obtiene para dar a ese paciente un tratamiento óptimo y no debe utilizarse para ningún otro fin, excepto, llegado el caso, para proyectos de investigación estrictamente controlados para los que se puedan necesitar determinados datos de un conjunto concreto de pacientes. En todos los demás casos, la información se ha de proteger contra los barridos de datos.

Hay dos grandes tipos de control de acceso: físico y lógico. El control de acceso físico limita el acceso a campus, edificios, salas y bienes de tecnología de la información (TI) físicos. El control de acceso lógico limita las conexiones a las redes informáticas, los ficheros de sistema y los datos. En esta Recomendación sólo se considera el control de acceso lógico.

El control de acceso comprende la autentificación por el proveedor de servicios de los solicitantes de acceso. En esta Recomendación se supone la utilización de firmas digitales y la existencia de una infraestructura de clave pública (PKI).

La Recomendación UIT-T X.1080.0 puede citarse como referencia en otras especificaciones sobre telebiometría.

En el Anexo A, que es parte integrante de esta Recomendación, se especifica la atribución de los identificadores de objeto utilizados en las Recomendaciones UIT-T de la Serie X.1080.

En el Anexo B, que es parte integrante de esta Recomendación, se presenta el perfil telebiométrico de la sintaxis de mensaje criptográfico (CMS), tal y como se define en IETF RFC 5652, que se utiliza en esta Recomendación. También podrá citarse como referencia en otras especificaciones sobre telebiometría.

En el Anexo C, que es parte integrante de esta Recomendación, se presenta una especificación formal de la aseveración de privilegios y los protocolos de asignación en forma de módulo de notación de sintaxis abstracta uno (ASN.1).

En el Apéndice I, que no forma parte integrante de esta Recomendación, se presenta una especificación informal del perfil CMS en forma de módulo ASN.1.

Recomendación UIT-T X.1080.0

Control de acceso para la protección de datos telebiométricos

1 Alcance

En esta Recomendación se especifica cómo proteger la privacidad en el entorno de la telebiometría utilizando un control de acceso telebiométrico (ACT) basado en la privacidad. Si bien en esta Recomendación no se identifican todos los tipos de información posibles, entra dentro de su alcance facilitar herramientas generales para el tratamiento de todo tipo de información de manera segura, lo que comprende la definición de un protocolo para la asignación de privilegios y de un protocolo para acceder a la información utilizando una aseveración de privilegios. Esta Recomendación no contiene requisitos de obligado cumplimiento, sino directrices.

Queda fuera del alcance de la presente Recomendación:

– La protección física de la información.

– El acceso no autorizado por el personal operativo que se ocupa del mantenimiento del sistema de seguridad y, por tanto, tiene la posibilidad de esquivar la seguridad.

2 Referencias

Las siguientes Recomendaciones UIT-T y demás referencias contienen disposiciones que, por referencia a las mismas en este texto, constituyen disposiciones de esta Recomendación. En la fecha de publicación, las ediciones citadas estaban en vigor. Todas las Recomendaciones y demás referencias están sujetas a revisión, por lo que se alienta a los usuarios de esta Recomendación a que consideren la posibilidad de aplicar la edición más reciente de las Recomendaciones y demás referencias que se indican a continuación. Se publica periódicamente una lista de las Recomendaciones UIT-T vigentes. La referencia a un documento en el marco de esta Recomendación no confiere al mismo, como documento autónomo, el rango de Recomendación.

[Serie UIT-T X.500] Recomendaciones UIT-T de la Serie X.5xx (2016) | ISO/CEI 9594-x, *Tecnología de la información – Interconexión de sistemas abiertos – El directorio*.

[UIT-T X.501] Recomendación UIT-T X.501 (2016) | ISO/CEI 9594-2, *Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Modelos*.

[UIT-T X.509] Recomendación UIT-T X.509 (2016) | ISO/CEI 9594-8, *Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Marcos para certificados de claves públicas y atributos*.

[UIT-T X.520] Recomendación UIT-T X.520 (2016) | ISO/CEI 9594-6, *Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Tipos de atributos seleccionados*.

[UIT-T X.521] Recomendación UIT-T X.521 (2016) | ISO/CEI 9594-7, *Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Clases de objeto seleccionadas*.

[UIT-T X.680] Recomendación UIT-T X.680 (2015) | ISO/CEI 8824-1, *Tecnología de la información – Notación de sintaxis abstracta uno (ASN.1): Especificación de la notación básica*.

[UIT-T X.681] Recomendación UIT-T X.681 (2015) | ISO/CEI 8824-2, *Tecnología de la información – Notación de sintaxis abstracta uno (ASN.1): Especificación de objetos de información*.

[UIT-T X.682] Recomendación UIT-T X.682 (2015) | ISO/CEI 8824-3, *Tecnología de la información – Notación de sintaxis abstracta uno (ASN.1): Especificación de constricciones*.

[UIT-T X.683] Recomendación UIT-T X.683 (2015) | ISO/CEI 8824-4, *Tecnología de la información – Notación de sintaxis abstracta uno (ASN.1): Parametrización de las especificaciones de la notación de sintaxis abstracta uno*.

[UIT-T X.690] Recomendación UIT-T X.690 (2015) | ISO/CEI 8825-1, *Tecnología de la información – Reglas de codificación de notación de sintaxis abstracta uno: Especificación de las reglas de codificación básica (BER), de las reglas de codificación canónica (CER) y de las reglas de codificación distinguida (DER)*.

[UIT-T X.1080.1] Recomendación UIT-T X.1080.1 (2011), *Telebiometría – Cibersalud y telemedicinas mundiales – Protocolo de telecomunicaciones genérico*.

[UIT-T X.1081] Recomendación UIT-T X.1081 (2011), *Telebiometría – El modelo telebiométrico multimodal – Marco para la especificación de los aspectos de la telebiometría relativos a protección y seguridad*.

[IETF RFC 2631] IETF RFC 2631 (1999), *Diffie-Hellman Key Agreement Method*.

[IETF RFC 3185] IETF RFC 3185 (2001), *Reuse of CMS Content Encryption Keys*.

[IETF RFC 5083] IETF RFC 5083 (2007), *Cryptographic Message Syntax (CMS) - Authenticated-Enveloped-Data Content Type*.

[IETF RFC 5652] IETF RFC 5652 (2009), *Cryptographic Message Syntax (CMS)*.

[IETF RFC 5753] IETF RFC 5753 (2010), *Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)*.

[IETF RFC 5911] IETF RFC 5911 (2010), *New ASN.1 Modules for Cryptographic Message Syntax (CMS) and S/MIME*.

[IETF RFC 6268] IETF RFC 6268 (2011), *Additional New ASN.1 Modules for the Cryptographic Message Syntax (CMS) and the Public Key Infrastructure Using X.509 (PKIX)*.

3 Definiciones

3.1 Términos definidos en otros documentos

En la presente Recomendación se utilizan los siguientes términos definidos en otros documentos:

3.1.1 certificado de atributo [UIT-T X.509]: Estructura de datos, firmada digitalmente por una autoridad de atributo, que vincula algunos valores de atributo con información de identificación de su titular.

3.1.2 tipo de atributo [UIT-T X.501]: El componente de un atributo que indica la clase de información dada por ese atributo.

3.1.3 valor de atributo [UIT-T X.501]: Un caso particular de la clase de información indicada por un tipo de atributo.

3.1.4 privilegio [UIT-T X.509]: Atributo o propiedad asignado a una entidad por una autoridad.

3.1.5 titular de privilegios [UIT-T X.509]: Entidad a la que se ha asignado un privilegio. El titular de un privilegio puede aseverar su privilegio para un fin concreto.

3.1.6 verificador de privilegios [UIT-T X.509]: Entidad que verifica certificados a partir de una política de privilegios.

3.1.7 fuente de autoridad (SOA, *source of authority*) [UIT-T X.509]: Autoridad de atributo en la que confía un verificador de privilegios para un recurso determinado como la autoridad última en asignar un conjunto de privilegios para aseverar ese recurso.

3.2 Términos definidos en la presente Recomendación

En la presente Recomendación se definen los siguientes términos:

3.2.1 control de acceso: Técnica de seguridad utilizada para regular quién puede hacer qué con los recursos de información en un entorno informático.

3.2.2 servicio de acceso: Servicio prestado por un proveedor de servicios para la ejecución de una transacción concreta.

3.2.3 solicitante de acceso: Titular de privilegios que accede a un servicio de acceso concreto utilizando su privilegio.

3.2.4 atributo: Elemento de información de un tipo concreto asociado a un objeto. La información asociada a un objeto se compone de atributos.

3.2.5 dominio de protección de datos: Dominio donde la información que se ha de proteger está sujeta a un único componente gestión.

3.2.6 nombre distinguido: Nombre que identifica un objeto único dentro de un contexto específico y que está formado por uno o más componentes de nombre que reflejan la posición del objeto dentro de una jerarquía de objetos.

3.2.7 objeto: Persona, departamento, profesional o cualquier otro tipo de objeto sobre el que hay información y que se identifica por un nombre distinguido.

3.2.8 clase de objeto: Familia identificada de entidades que comparten ciertas características.

3.2.9 operación: Interacción compuesta por una solicitud y una respuesta entre un solicitante de acceso y un proveedor de servicios con un objetivo concreto.

3.2.10 especificación: Recomendación UIT-T, norma internacional o cualquier especificación elaborada por un organismo de normalización reconocido.

4 Abreviaturas y acrónimos

En esta Recomendación se utilizan las siguientes abreviaturas y acrónimos:

AA Autoridad de atributo (*Attribute Authority)*

ABAC Control de acceso por atributos (*Attribute Based Access Control)*

ACL Lista de control de acceso (*Access Control List)*

ACT Control de acceso telebiométrico (*Access Control for Telebiometrics)*

AES Norma de encriptación avanzada (*Advanced Encryption Standard)*

ASN.1 Notación de sintaxis abstracta uno (*Abstract Syntax Notation One)*

BER Reglas de codificación básica (*Basic Encoding Rules)*

CA Autoridad de certificación (*Certification Authority)*

CEK Clave de encriptación de contenido (*Content Encryption Key)*

CMS Sintaxis de mensaje criptográfico (*Cryptographic Message Syntax)*

DER Reglas de codificación distinguida (*Distinguished Encoding Rules)*

DH Diffie-Hellman

ECC	Criptografía de curva elíptica (*Elliptic Curve Cryptography)*
ECDH	Diffie-Hellman de curva elíptica (*Elliptic Curve Diffie-Hellman)*
GCM	Modo Galois/Counter (*Galois/Counter Mode)*
KEK	Clave de encriptación por claves (*Key-Encryption Key)*
LDAP	Protocolo ligero de acceso al directorio (*Lightweight Directory Access Protocol)*
MAC	Código de autentificación de mensaje (*Message Authentication Code)*
PDU	Unidad de datos de protocolo (*Protocol Data Unit)*
PKI	Infraestructura de clave pública (*Public-Key Infrastructure)*
PMI	Infraestructura de gestión de privilegios (*Privilege Management Infrastructure)*
SDO	Organismo de normalización (Standards Developing Organization)
SOA	Fuente de autoridad (*Source of Authority)*

5 Convenios

En esta Recomendación se presenta la notación de sintaxis abstracta uno (ASN.1) con el tipo **`courier new typeface en negrita`**. Cuando en el texto normal se hace referencia a tipos y valores ASN.1 éstos se distinguen del texto normal presentándolos en tipo Courier New typeface en negrita.

Si los puntos de una lista están numerados, (en lugar de utilizar símbolos como "–" o letras), serán considerados etapas de un procedimiento.

6 Conceptos y modelos básicos

6.1 Protección en un único dominio de protección de datos

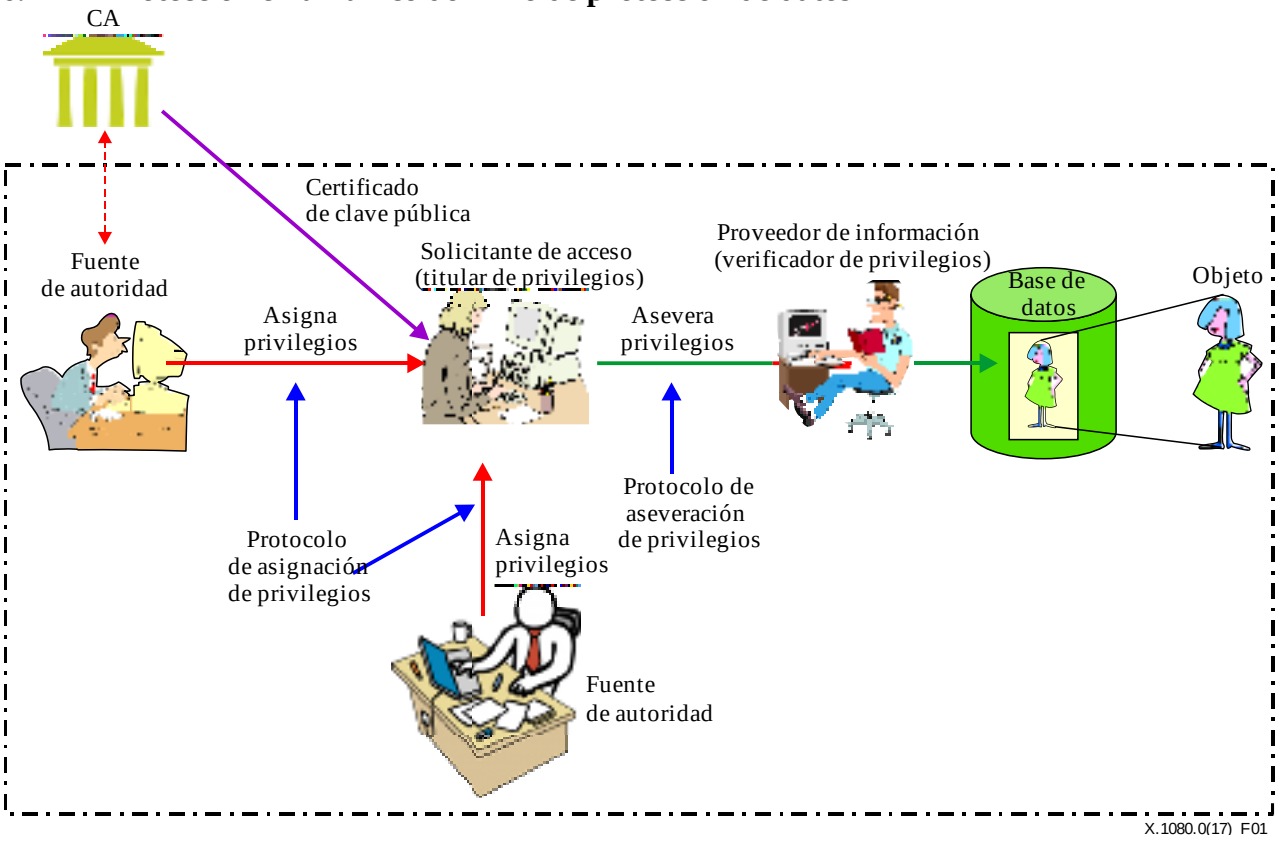

Figura 1 – Modelo con un solo dominio de protección de datos

En la Figura 1 se muestran las distintas entidades y protocolos que participan en el entorno de control de acceso dentro de un solo dominio de protección de datos. Un dominio de protección de datos está formado por todas las entidades que participan en la protección y que están sometidas a una gestión común.

Una entidad, denominada solicitante de acceso, puede tener una función laboral que necesita un permiso para realizar operaciones sobre las informaciones acerca de un objeto que posee un proveedor de información. Por ejemplo, el médico responsable de un paciente necesita acceder a su expediente médico para darle el tratamiento óptimo, mientras que los demás médicos, que no se ocupan de ese paciente, no necesitan esa información.

Sólo deberá darse al solicitante de acceso permiso para realizar una operación concreta sobre la información si ese solicitante tiene el *derecho* y la *necesidad genuina* de realizar esa operación, es decir, que se le ha asignado el *privilegio* necesario.

Queda fuera del alcance de esta Recomendación especificar en qué condiciones se asignan los privilegios a las entidades. Sólo corresponde presentar las herramientas necesarias para gestionar los privilegios de manera segura.

Un dominio de protección de datos debe crear una o más autoridades para asignar privilegios a las entidades. Se utiliza aquí el término fuente de autoridad (SOA) definido en [UIT-T X.509] para designar a la más alta autoridad de asignación de privilegios en una esfera específica dentro de un dominio de protección de datos.

En esta Recomendación se utilizan certificados para asignar privilegios a los solicitantes de acceso. Los privilegios pueden estar incluidos en el componente **`attributes`** de los certificados de atributo o en la extensión **`subjectDirectoryAttributes`** de los certificados de clave pública. Los privilegios de carácter permanente generalmente se incluirán en certificados de clave pública, mientras que los de orden temporal se concederán en certificados de atributo.

En la Figura 1 se ilustra la relación entre los distintos componentes dentro de un solo dominio de protección de datos. Se muestran dos SOA, cada una de ellas responsable de la asignación de privilegios a los titulares de privilegios para distintos fines.

El solicitante de acceso puede necesitar algunos privilegios para las operaciones cotidianas, por lo que son permanentes, mientras que hay otros privilegios que se asignan para afrontar situaciones especiales, por lo que son de carácter temporal.

Cuando se asignan privilegios a un solicitante de acceso, éste se convierte en titular de privilegios y puede utilizarlos para acceder a la información a través del protocolo de aseveración de privilegios. El verificador de privilegios, que representa al proveedor de información, verifica al privilegio aseverado antes de otorgar acceso a la información.

Una SOA puede remitir los privilegios por medios locales o integrándolos en un certificado de atributo firmado en el protocolo de asignación de privilegios, como se muestra en la Figura 1.

6.2 Dominio de protección de datos cruzados

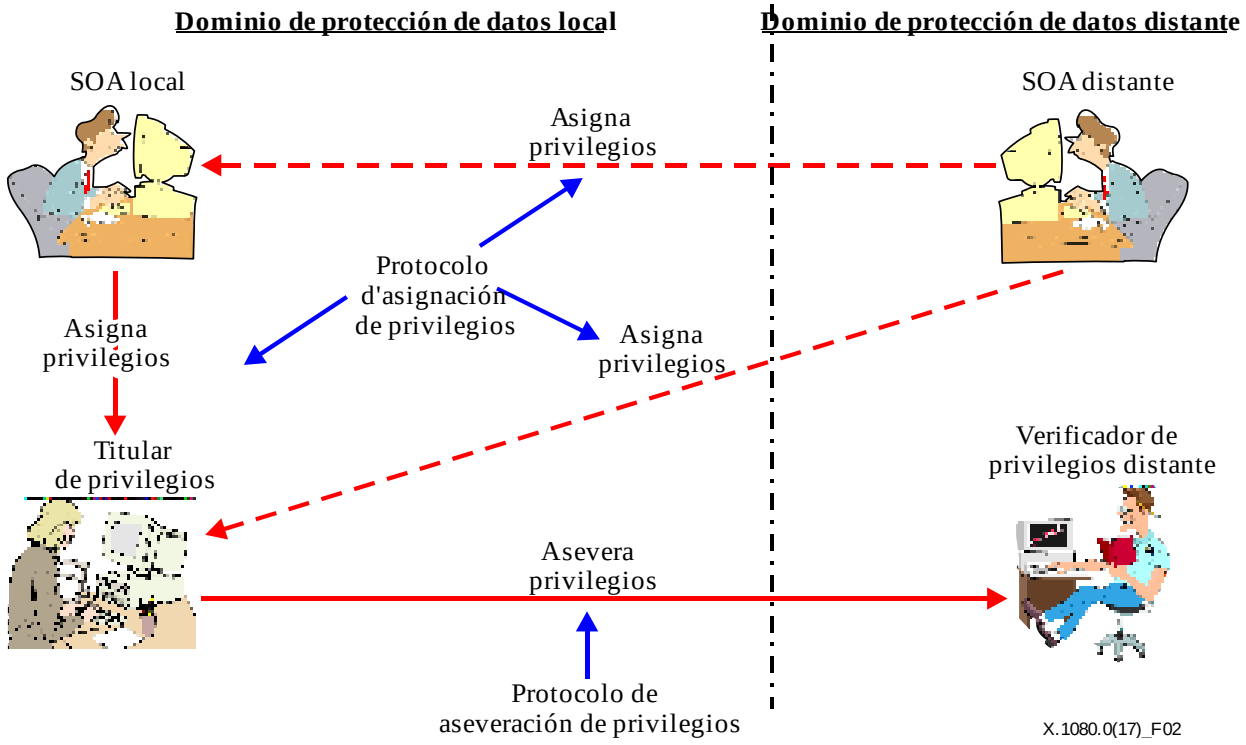

Figura 2 – Modelo de dominio de protección de datos cruzados

En la Figura 2 se muestra el caso en que el solicitante de acceso necesita acceder a información de otro dominio de protección de datos, por ejemplo, para tratar a un paciente cuya información médica vital se encuentra en otro dominio de protección de datos.

En este entorno, el solicitante, directamente o a través de una SOA local, solicita acceder a una información específica sobre un objeto concreto. Cómo se remite esa solicitud queda fuera del alcance de esta Recomendación, pero puede ser por correo-e o por teléfono.

La SOA distante genera un certificado de atributo con un atributo de tipo **`accessService`** que contiene el privilegio necesario. Ese certificado de atributo está firmado por la SOA distante. El **`holder`** del certificado de atributo puede ser la SOA local o la entidad que necesita el privilegio.

Si el privilegio se asigna a la SOA local, ésta:

a) desencripta el contenido, de ser necesario;

b) verifica la firma del contenido;

c) extrae el certificado de atributo del contenido;

d) valida el certificado de atributo extraído;

e) crea un nuevo certificado de atributo con información de privilegios obtenida del certificado de atributo recibido e indicando que el titular es la entidad que necesita el privilegio;

f) este certificado de atributo, junto con el recibido de la SOA distante, se remite entonces a la entidad que necesita el privilegio (es decir, el titular de privilegios).

6.3 Modelo orientado al servicio

El control de acceso telebiométrico se orienta al servicio en el sentido de que el solicitante de acceso puede invocar una serie de funciones distintas, denominadas *servicios de acceso*. Un servicio de acceso es una función ofrecida por un proveedor de servicios capaz de ejecutar ese servicio de acceso en particular. Un ejemplo de servicio de acceso es la capacidad para manipular la información de los

pacientes de un hospital. Se utiliza un identificador de objeto para identificar cada tipo de servicio de acceso concreto. En esta Recomendación no se definen servicios de acceso específicos, pero se ofrecen las herramientas para crear servicios de acceso. En especificaciones derivadas podrán definirse los servicios de acceso pertinentes a su alcance.

Aunque un solicitante de acceso pueda acceder a un servicio concreto, es posible que no tenga los privilegios que abarquen todos los aspectos de ese servicio.

6.4 El modelo objeto y atributo

El modelo de información definido en [UIT-T X.501] está diseñado para manejar distintas estructuras de datos. En esta Recomendación se utiliza ese modelo de información para el control de acceso.

En el modelo UIT-T X.501, los objetos que se han de proteger se organizan en clases de objetos, donde los objetos que pertenecen a una clase tienen características comunes pertinentes dentro de un contexto específico. Una clase de objetos se identifica mediante un identificador de objeto. En [UIT-T 1080.1] se amplía este concepto mediante la asignación de identificadores de objeto a un grupo de clases de objetos para categorías pertinentes a la telebiometría.

En [UIT-T X.521] se define también un conjunto de clases de objetos utilizables de orden general. Cuando se necesite una clase de objetos que no esté directamente relacionada con la telebiometría, se utilizará, siempre que sea posible, una clase de objetos ya definida.

Los objetos individuales se identifican mediante nombres distinguidos, como se define en [UIT-T X.501]. Un nombre distinguido está formado por uno o más componentes de nombre que reflejan su posición dentro de una jerarquía de objetos.

La información asociada a un objeto se modeliza como un conjunto de atributos. Los atributos con características comunes constituyen un tipo de atributo. Los tipos de atributo se identifican mediante identificadores de objeto. Un atributo es una secuencia de identificadores de objeto que identifican el tipo de atributo y uno o más valores de ese tipo. Un objeto puede tener sólo un atributo de un tipo concreto. En [UIT-T X.520] se define un conjunto de tipos de atributo utilizables de carácter general. Siempre que sea posible se utilizarán los tipos de atributo ya definidos. Podrán definirse, en función de las necesidades de especificaciones individuales, tipos de atributo adicionales.

Cada aplicación concreta de esta Recomendación deberá definir la correspondencia entre este modelo de información y la estructura básica de datos real. Esa correspondencia será fácil de establecer si la información se mantiene en un directorio de protocolo ligero de acceso al directorio (LDAP) o en un directorio seguro basado en las especificaciones de la [Serie UIT-T X.500].

6.5 Principios básicos del control de acceso

El objetivo de esta cláusula es presentar brevemente los principios generales aplicables al establecimiento de un control de acceso telebiométrico. El control de acceso tradicional se ocupa principalmente de dar o denegar el acceso a los datos en función de parámetros permanentes. En esta Recomendación se amplía el concepto considerando también la necesidad de conocer, lo que se consigue adoptando el enfoque de servicio expuesto en la cláusula 6.3. Para realizar una operación concreta, el solicitante de acceso debe tener el privilegio que permite acceder al servicio de acceso pertinente y el privilegio que permite acceder a la información necesaria.

Este privilegio comprende el acceso a objetos de una o más clases de objetos, posiblemente limitado a algunos objetos nombrados. El tipo de operación que puede realizarse puede ser distinto para distintas clases de objetos y objetos nombrados.

Las operaciones en los objetos pueden incluir operaciones en atributos individuales y en valores de atributos. Las operaciones permitidas pueden diferir de un tipo de atributo a otro.

6.6 Relación con otros esquemas de control de acceso

Existen distintos tipos de control de acceso para cubrir las diferentes necesidades de control de acceso. A continuación se describen brevemente distintos esquemas de control de acceso y su relación con el ACT.

6.6.1 Control de acceso básico definido en [UIT-T X.501]

El control de acceso básico definido en [UIT-T X.501] se utiliza para proteger la información de un directorio, como se define en la [Serie UIT-T X.500]. Se prevé la existencia de listas complejas de control de acceso donde se especifica detalladamente qué usuarios tienen derecho al acceso y cómo se les concede ese acceso. La diferencia con esta Recomendación reside en que sólo se prevé el derecho a saber, pero no la necesidad de saber.

6.6.2 Control de acceso basado en reglas

Tanto en [UIT-T X.501] como en [b-UIT-T X.841] se define un tipo de control de acceso basado en reglas. En este tipo de control de acceso, los datos que se han de proteger se etiquetan con información sobre qué protección se requiere, mientras que los usuarios que acceden tienen información certificada asociada en una solicitud de acceso que especifica su nivel de acreditación con respecto al acceso a cierta información. Este concepto difiere de esta Recomendación en que se exige el etiquetado de los datos almacenados y sólo se contempla el derecho a saber, pero no la necesidad de conocer.

6.6.3 Control de acceso basado en funciones definido para la red eléctrica inteligente

El control de acceso basado en funciones, especificado en [b-CEI 62351-8], se centra en los usuarios y en el trabajo de éstos. Una función es una serie de derechos con respecto a objetos (acciones que pueden realizarse en determinados objetivos). Un usuario puede tener una o más funciones. Para el control de acceso, una función es un tipo de intermediario que reduce la cantidad de información necesaria en la lista de control de acceso (ACL) reduciendo la granularidad de la información de control de acceso. Los permisos de acceso a objetos del sistema no se enumeran en una lista para cada usuario individualmente, sino que se otorgan funciones a los usuarios y se describen sólo una vez los derechos asociados a cada función.

6.6.4 Control de acceso basado en atributos

El control de acceso basado en atributos (ABAC) es un modelo de control de acceso lógico distinguible porque controla el acceso a los objetos evaluando en función de unas reglas establecidas los atributos de las entidades (sujetos y objetos), las operaciones y el entorno en que se plantea la solicitud. Los sistemas ABAC son capaces de aplicar tanto el concepto de control de acceso discrecional, como el de control de acceso obligatorio. ABAC permite ejercer un control de acceso preciso, por lo que las decisiones de control de acceso pueden tener un mayor número de entradas discretas, pudiendo así realizar un conjunto mayor de combinaciones posibles de esas variables para dar lugar a un mayor y más definitivo conjunto de reglas posibles con las que expresar las políticas.

En [b-NIST 800-162] puede encontrarse una buena introducción a ABAC.

6.7 Aspectos generales de los protocolos

6.7.1 Protocolo de evaluación de privilegios

El protocolo de evaluación de privilegios comprende una serie de tipos de contenido en sintaxis de mensaje criptográfico (CMS). Cada tipo de acceso exige un tipo de contenido solicitud y un tipo de contenido resultado.

Las instancias de tipo de contenido se transmiten utilizando CMS con el perfil indicado en el Anexo B. en el Anexo C se presenta un módulo ASN.1 formal.

6.7.2 Protocolo de asignación de privilegios

El protocolo de asignación de privilegios se emplea para transmitir certificados de atributo entre SOA o entre SOA y el titular de privilegios.

Para este protocolo se define un único par de tipos de contenido: un tipo de contenido para remitir el privilegio en forma de certificado de atributo y un tipo de contenido para dar cuenta del resultado.

Las instancias de tipo de contenido se transmiten utilizando CMS con el perfil indicado en el Anexo B. En el Anexo C se presenta un módulo ASN.1 formal.

6.8 Utilización de CMS

En el Anexo B se presenta un perfil para la utilización telebiométrica de CMS.

Para garantizar que la fuente de información se autentifica adecuadamente, los tipos de contenido definidos en esta Recomendación se encapsularán en una instancia del tipo de contenido **`signedData`**. Dado que puede transmitirse información sensible, se recomienda encapsularla también en una instancia del tipo de contenido **`envelopedData`**. El tipo de contenido ct-**`autEnvelopedData`** no se utiliza en esta Recomendación.

6.9 Consideraciones sobre los certificados de clave pública

Un certificado de atributo deberá estar firmado por el emisor con su clave privada y esa firma se verificará utilizando el correspondiente certificado de clave pública expedido al emisor del certificado de atributo.

En principio, la misma clave privada podría utilizarse para firmar el mensaje CMS utilizando el tipo de contenido **`signedData`**, lo que facilitaría el proceso de verificación. Sin embargo, la utilización de la misma clave privada para distintos fines plantea problemas de seguridad. Puede considerarse la posibilidad de utilizar distintas claves privadas para cada uno de los fines, pero no es una opción obligatoria en esta Recomendación.

7 Configuración de la información de privilegios

7.1 Utilización de certificados de atributo

[UIT-T X.509] permite que la información de privilegios se transporte en certificados de clave pública y en certificados de atributo. En ambos casos, las especificaciones de privilegios van en atributos, como se define en [UIT-T X.501]. Los tipos de atributo para este fin son distintos de los tipos de atributo utilizados para modelizar información sobre objetos. Mientras que los tipos de atributo definidos para transportar información sobre un objeto deben ser lo más simples posible, los tipos de atributo para transportar información de privilegios serán, por su propia naturaleza, bastante complejos.

Así, al utilizar un certificado de atributo para transportar información de privilegios:

a) el componente **`holder`** identifica la identidad a que se asignan los privilegios. Cuando el titular de privilegios presente este certificado de atributo que contiene los privilegios a un verificador de privilegios, el verificador de privilegios autentificará al solicitante de acceso para verificar que realmente es el titular de privilegios del certificado de atributo;

b) el componente **`issuer`** contiene el nombre de la SOA o el nombre de una autoridad de atributo (AA) en la que se ha delegado la asignación de privilegios. El verificador de privilegios también obtendrá y validará el certificado de clave pública del emisor para validar la firma en el certificado de atributo;

c) el componente **`attributes`** contiene un atributo de tipo **`accessService`**, como se define en la cláusula 7.3.

La SOA que ha aprobado el privilegio o la AA en que se ha delegado la emisión firmarán el certificado de atributo.

La validación resultará más sencilla si el titular y el emisor poseen certificados de clave pública emitidos por la misma autoridad de certificación (CA).

7.2 Utilización de certificados de clave pública

Al utilizar un certificado de clave pública para transportar información de privilegios:

a) el componente **subject** identifica al solicitante de acceso al que se han asignado los privilegios. Cuando un solicitante de acceso presente este certificado de clave pública que contiene los privilegios a un verificador de privilegios, el verificador de privilegios autentificará al solicitante de acceso;

b) el componente **issuer** contiene el nombre distinguido de la CA responsable de la emisión del certificado de clave pública;

c) la extensión **subjectDirectoryAttributes** contiene una instancia del tipo de atributo **accessService** (véase la cláusula 7.3).

7.3 Tipo de atributo accessService

7.3.1 Sintaxis del atributo accessService

Se supone que se incluirá un atributo de tipo **accessService** en el componente **attributes** de un certificado de atributo o en la extensión **subjectDirectoryAttributes** de un certificado de clave pública. El atributo tiene la siguiente sintaxis:

```
accessService ATTRIBUTE ::= {
  WITH SYNTAX  AccessService
  ID           id-at-accessService }
```

Un atributo de tipo **AccessService** da información de privilegios a fin de que el verificador de privilegios pueda determinar si se ha de aceptar o no la solicitud de acceso. Se trata de un tipo de atributo multivalor que permite incluir en el mismo atributo múltiples tipos de servicios de acceso y los permisos asociados.

Una entidad no puede utilizar un servicio si no está incluido en este atributo.

La sintaxis del tipo de atributo **accessService** es la siguiente:

```
AccessService ::= SEQUENCE {
  serviceId          OBJECT IDENTIFIER,
  objectDef          SEQUENCE SIZE (1..MAX) OF ObjectSel,
  ... }
```

Los componentes de un valor del tipo de datos **AccessService** son:

a) el componente **serviceId**, que identifica el tipo de servicio de acceso para el que el solicitante de acceso tiene privilegios;

b) el componente **objectDef**, que especifica las clases de objetos para las que se han asignado privilegios al titular del certificado de atributos para un tipo de servicio concreto. Contendrá un elemento para cada clase de objeto para la que se han asignado privilegios. El titular de privilegios no tiene, para este servicio de acceso, acceso a las clases de objetos no enumeradas en el componente **objectDef**.

7.3.2 Selección de objetos

La selección de objetos para los que el solicitante de acceso tiene privilegios se especifica en una instancia del tipo de datos **ObjectSel**.

```
ObjectSel ::= SEQUENCE {
  objecClass          OBJECT-CLASS.&id,
  objSelect           CHOICE {
    allObj         [0]  TargetSelect,
    objectNames    [1]  SEQUENCE SIZE (1..MAX) OF SEQUENCE {
      object              CHOICE {
        names          [1]  SEQUENCE SIZE (1..MAX) OF DistinguishedName,
        subtree        [2]  DistinguishedName,
        ... },
      select           TargetSelect,
      ... },
    ... },
  ... }
```

Un valor del tipo de datos **ObjectSel** especifica el privilegio para cada clase de objetos para las que se han asignado privilegios para el tipo de servicio de acceso en cuestión. Tiene los siguientes componentes:

– el componente **objectClass** especifica la clase de objeto para la que se han asignado privilegios;

– el componente **objSelect** especifica los objetos de la clase de objetos para la que se han asignado privilegios. Hay dos alternativas:

  a) se opta por la alternativa **allObj** si los privilegios asignados se aplican por igual a todos los objetos de la clase. Contiene una instancia del tipo de datos **TargetSelect**;

  b) se opta por la alternativa **objectNames** si los privilegios sólo se aplican a objetos seleccionados de la clase de objetos identificada. Puede estar formado por múltiples elementos, donde cada uno de ellos está compuesto por:

    i) el componente **object**, que especifica uno o más objetos para los que se han asignado privilegios. Tiene las siguientes alternativas:

      • la alternativa **names** especifica el nombre de uno o más objetos a los que se aplican los privilegios;

      • la alternativa **subtree** se escoge para grupos de objetos, donde cada objeto tiene un nombre distinguido igual al nombre distinguido que contiene esta alternativa o sus componentes de nombre iniciales son iguales a ese nombre distinguido;

    ii) el componente de selección, que contiene una instancia del tipo de datos **TargetSelect**.

El tipo de datos **TargetSelect** tiene la siguiente sintaxis:

```
TargetSelect ::= SEQUENCE {
  objOper   ObjectOperations OPTIONAL,
  attrSel   AttributeSel     OPTIONAL,
  ... }
  (WITH COMPONENTS {..., objOper  PRESENT } |
   WITH COMPONENTS {..., attrSel  PRESENT } )
```

El tipo de datos **TargetSelect** tiene los dos siguientes componentes opcionales, de los cuales, al menos uno debe estar presente:

a) cuando está presente, el componente **objOper** especifica las operaciones permitidas en los objetos de la clase de objetos o en los objetos seleccionados. Si no está presente, no se permiten operaciones en los objetos en conjunto;

b) cuando está presente, el componente **attrSel** contiene un valor del tipo de datos **AttributeSel** (véase la cláusula 7.3.3). Si no está presente, no se permiten operaciones en los atributos de los objetos.

7.3.3 Selección de tipos de atributo

```
AttributeSel ::= SEQUENCE {
  attSelect           CHOICE {
    allAttr        [0]  SEQUENCE {
      attrOper1      [0]  AttributeOperations OPTIONAL,
      ... },
    attributes     [1]  SEQUENCE SIZE (1..MAX) OF SEQUENCE {
      select              SEQUENCE SIZE (1..MAX) OF ATTRIBUTE.&id,
      attrOper2      [0]  AttributeOperations OPTIONAL,
      ... },
    ... },
  ... }
```

El tipo de datos **AttributeSel** especifica los tipos de atributo a los que se aplican los privilegios. Tiene los siguientes componentes:

– El componente **attSelect** tiene dos alternativas:

  a) la alternativa **allAttr** se escoge si el privilegio se aplica a todos los atributos del/de los objeto(s). Está compuesto por:

    i) el componente **attrOper1**, que especifica las operaciones que se pueden realizar en los atributos;

  b) se opta por la alternativa **attributes** si los privilegios se aplican sólo a algunos atributos del/de los objeto(s). El solicitante de acceso no tiene privilegios para los tipos de atributo no enumerados y no se le darán a conocer los tipos de atributo no enumerados. Esta alternativa tiene los siguientes componentes:

    i) el componente **select**, que especifica uno o más tipos de atributo a los que se aplican los privilegios;

    ii) el componente **attrOper2**, que especifica las operaciones que pueden realizarse en los atributos.

7.4 Operaciones en objetos en conjunto

Para especificar las operaciones permitidas en un objeto se utilizan los siguientes tipos de datos:

```
ObjectOperations ::= BIT STRING {
  read            (0),
  add             (1),
  modify          (2),
  delete          (3),
  rename          (4),
  discloseOnError (5) }
```

El permiso **read** se configurará para el solicitante de acceso con permiso para leer la información de un objeto.

El permiso **add** se configurará para el solicitante de acceso con permiso para añadir nuevos objetos de la clase de objetos específica. Se necesita el permiso **add** para todos los objetos de una clase específica. Se necesitará un permiso **add** para cada tipo de atributo que se vaya a añadir al objeto (véase la cláusula 7.5).

El permiso **modify** se configurará para el solicitante de acceso con permiso para modificar un objeto existente. El solicitante de acceso obtendrá el permiso **modify** para la clase de objeto en su conjunto o para el/los objeto(s) nombrado(s) que se vaya(n) a modificar. Si el solicitante de acceso añade atributos, deberá tener permisos **add** para los tipos de atributo en cuestión. Si el solicitante de acceso suprime atributos, deberá tener el permiso **delete** para los tipos de atributo en cuestión. Si el solicitante de acceso modifica atributos, deberá tener el permiso **modify** para los tipos de atributo en cuestión. Si el solicitante de acceso suprime atributos, deberá tener el permiso **deleteValue** para los

tipos de atributo en cuestión. Si el solicitante de acceso sustituye atributos, deberá tener el permiso **replaceAttribute** para los tipos de atributo en cuestión.

El permiso **delete** se configurará para el solicitante de acceso con permiso para suprimir un objeto existente. El solicitante de acceso obtendrá el permiso **delete** para la clase de objetos en su conjunto o para el/los objeto(s) nombrado(s) que se vaya(n) a suprimir.

El permiso **rename** se configurará para el solicitante de acceso con permiso para renombrar objetos existentes. El solicitante de acceso obtendrá el permiso **rename** para la clase de objetos en su conjunto o para el/los objeto(s) nombrado(s) que se vaya(n) a renombrar.

El **discloseOnError** se configurará para el solicitante de acceso con permiso para conocer la existencia del objeto cuando la operación falle.

7.5 Operaciones en atributos

```
AttributeOperations ::= BIT STRING {
  read              (0),
  compare           (1),
  add               (2),
  modify            (3),
  delete            (4),
  deleteValue       (5),
  replaceAttribute  (6),
  discloseOnError   (7) }
```

El permiso **read** se configurará para cada uno de los tipos de atributos deseados cuya lectura se permita al solicitante de acceso. El solicitante de acceso obtendrá el permiso **read** para la clase de objetos pertinente en su conjunto o para los objetos nombrados pertinentes. Además, deberá tener el permiso **read** para todos los atributos de esas clases de objetos o acceso a los tipos de atributos pertinentes.

El permiso **compare** se configurará para el solicitante de acceso con permiso para comparar uno o más atributos. El privilegio contendrá el permiso **read** para la clase de objetos en su conjunto o para el/los objeto(s) nombrado(s). Además, deberá tener el permiso para comparar todos los atributos de las clases de objetos permitidas o el permiso para comparar los tipos de atributos pertinentes.

El permiso **add** se configurará para el solicitante de acceso con permiso para añadir uno o más atributos. El solicitante de acceso obtendrá el permiso **modify** para la clase de objetos en su conjunto o para el/los objeto(s) nombrado(s). Además, deberá tener el permiso **add** para los tipos de atributos pertinentes.

El permiso **modify** se configurará para el solicitante de acceso con permiso para modificar un atributo de un tipo específico. Además, el solicitante de acceso deberá tener el permiso **modify** para la clase de objetos en su conjunto o para el/los objeto(s) nombrado(s).

El permiso **delete** se configurará para el solicitante de acceso con permiso para suprimir uno o más atributos. Además, el solicitante de acceso deberá tener el permiso **modify** para la clase de objetos en su conjunto o para el/los objeto(s) nombrado(s).

El permiso **deleteValue** se configurará para el solicitante de acceso con permiso para suprimir uno o más valores de un atributo del tipo de atributo. Además, el solicitante de acceso deberá tener el permiso **modify** para la clase de objetos en su conjunto o para el/los objeto(s) nombrado(s).

El permiso **replaceAttribute** se configurará para el solicitante de acceso con permiso para sustituir un atributo de un tipo determinado por un atributo del mismo tipo. Además, el solicitante de acceso deberá tener el permiso **modify** para la clase de objetos en su conjunto o para el/los objeto(s) nombrado(s).

El **discloseOnError** se configurará para el solicitante de acceso con permiso para conocer la existencia de un atributo cuando la operación falle. Además, deberá tener el permiso **discloseOnError** para el objeto en su conjunto.

7.6 Tratamiento de errores

Pueden generarse errores como resultado de la utilización de CMS como se muestra en el Anexo A.5. Cuando se detecta un error no es necesario realizar verificaciones ulteriores. Los errores CMS se devuelven en el resultado de la solicitud de acceso.

También pueden generarse errores como resultado de la solicitud de acceso real.

Los errores se comunican mediante una instancia del tipo de datos **AccessdErr**, definido en la cláusula 8.10.

8 Protocolo de aseveración de privilegios

8.1 Aspectos generales

El siguiente objeto de información contiene todos los tipos de contenido definidos representados por los objetos de información definidos en esta Recomendación.

```
ActContentTypes CONTENT-TYPE ::= {
    privAssignRequest |
    privAssignResult |
    readRequest |
    readResult |
    compareRequest |
    compareResult |
    addRequest |
    addResult |
    deleteRequest |
    deleteResult |
    modifyRequest |
    modifyResult |
    renameRequest |
    renameResult,
    ... }
```

Los tipos de contenido especificados por el conjunto **ActContentTypes** constituyen el protocolo de aseveración de privilegios, que comprende una serie de diversas operaciones de acceso, que se especifican en las cláusulas 8.4 a 8.9.

8.2 Componentes de solicitud comunes

Todas las solicitudes contienen los siguientes componentes:

```
CommonReqComp ::= SEQUENCE {
  attrCerts [31]  AttributeCertificates OPTIONAL,
  serviceId [30]  OBJECT IDENTIFIER,
  invokId   [29]  INTEGER,
  ... }

AtributeCertificates ::= SEQUENCE SIZE (1..MAX) OF AttributeCertificate
```

Los parámetros de solicitud comunes son:

a) el componente **attrCert**, que, de estar presente, especifica el certificado de atributo o el trayecto de certificación de atributo que contiene el privilegio del solicitante de acceso. De no estar presente este componente, el privilegio para el solicitante de acceso se facilitará en el certificado de clave pública de la entidad extrema;

b) el componente **serviceId** especifica el tipo de servicio que se ha de invocar;

c) el componente **invokId** toma el valor cero para la primera operación invocada y se va incrementando en uno a cada operación invocada posterior. Su gama debe garantizar que no se reutiliza el mismo valor durante un lapso de tiempo importante de la comunicación entre dos entidades. Deberá poder detectar solicitudes ausentes y ataques de respuesta. El receptor de una solicitud utilizará el mismo valor en la respuesta para que el solicitante de acceso pueda emparejar los resultados con las solicitudes correspondientes.

8.3 Acceso a un servicio

Todos los tipos de operación especificados en las cláusulas 8.4 a 8.9 necesitan del acceso a un servicio concreto. El verificador de privilegios (receptor) comprobará que el privilegio asignado al solicitante de acceso en el certificado de atributo o certificado de clave pública asociados permite acceder al servicio solicitado.

Si el solicitante de acceso no tiene permiso para invocar el tipo de servicio solicitado o el proveedor de servicios no soporta el tipo de servicio solicitado, se devolverá un código de error **noSuchService**.

Si el solicitante de acceso tiene permiso para invocar el servicio, se verificará si la operación solicitada es coherente con el tipo de servicio y, de no serlo, se devolverá un código de error **invalidOperationForService**.

8.4 Operación leer

La operación leer comprende una solicitud de lectura y el resultado de lectura correspondiente.

Las solicitudes de lectura se transportan en una instancia del tipo de contenido **readRequest** y el resultado de lectura se transporta en una instancia del tipo de contenido **readResult**.

```
readRequest CONTENT-TYPE ::= {
              ReadRequest
IDENTIFIED BY id-readRequest
```

El solicitante de acceso utiliza el tipo de contenido **readRequest** para leer información de un objeto en particular.

```
ReadRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object    [1] DistinguishedName,
  selection [2] InformationSelection,
  ... }
```

El tipo de datos **ReadRequest** especifica la sintaxis del contenido real y tiene los siguientes componentes:

a) el componente **object** contiene el nombre distinguido del objeto sobre el que se solicita información;

b) el componente **selection** especifica el tipo de información que el solicitante de acceso solicita (véase la cláusula 8.11).

La solicitud de lectura fallará si en la solicitud se especifica un objeto desconocido y se devolverá un código de error **noSuchObject**.

La solicitud de lectura fallará si el solicitante de acceso no tiene el permiso **read** para el objeto de acuerdo con el privilegio asignado al solicitante de acceso. Si no se concede el permiso **read**, se devolverá un código de error **insufficientAccessRigth** en caso de que el solicitante de acceso disponga del permiso **discloseOnError** para ese objeto. En caso contrario, se devolverá un código de error **noSuchObject**.

Se necesita el permiso **read** para el tipo de atributo de cada uno de los atributos que se hayan de devolver. Si el solicitante de acceso no tiene el permiso **read** para un tipo de atributo en concreto, no se devolverá en el resultado un atributo de ese tipo. Si el resultado es que no se devuelven atributos, la solicitud fallará. Si el solicitante de acceso tiene el permiso **discloseOnError** para todas las solicitudes de atributo, se devolverá un código de error **insufficientAccessRigth**. En caso contrario, se devolverá un código de error **noInformation**.

```
readResult CONTENT-TYPE ::= {
              ReadResult
IDENTIFIED BY id-readResult }
```

El verificador de privilegios utilizará una instancia del tipo de contenido **readResult** para devolver la información solicitada o comunicar un error.

```
ReadResult ::= SEQUENCE {
  object    DistinguishedName,
  result    CHOICE {
    success   [0] ObjectInformation,
    failure   [1] AccessdErr,
    ... },
  ... }
```

El tipo de datos **ReadResult** especifica la sintaxis del contenido real y tiene los siguientes componentes:

a) el componente **object** contiene el nombre del objeto sobre el que se solicita información;

b) el componente resultado contiene el resultado de la solicitud de lectura y tiene dos alternativas:

   – si se ha de devolver la información, se escoge la alternativa **success**, que contendrá una instancia del tipo de datos **ObjectInformation** (véase la cláusula 8.12). La información devuelta es la intersección entre lo que solicita el solicitante de acceso y la información que está autorizado a extraer.

   – si se ha de devolver un error, se escoge la alternativa **failure**.

8.5 Operación comparar

La operación comparar está compuesta de una solicitud de comparación y una respuesta de comparación.

Las solicitudes de comparación se transportan en una instancia del tipo de contenido **compareRequest** y el resultado de comparación se transporta en una instancia del tipo de contenido **compareResult**.

```
compareRequest CONTENT-TYPE ::= {
              CompareRequest
IDENTIFIED BY id-compareRequest }
```

Se utiliza una instancia del tipo de contenido **compareRequest** para comparar un valor supuesto presentado de un tipo de atributo concreto con un valor de atributo del mismo tipo perteneciente a un objeto en particular.

```
CompareRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object    [1] DistinguishedName,
  purported [2] AttributeValueAssertion,
  ... }
```

El tipo de datos **CompareRequest** especifica el contenido real y, además de los definidos en la cláusula 8.2, tiene los siguientes componentes:

a) el componente **object** contiene el nombre distinguido del objeto cuyo valor de atributo se compara;

b) el componente **purported** contiene una combinación de tipo de atributo y de valor de atributo que se ha de comparar con un atributo del mismo tipo perteneciente al objeto en cuestión.

La solicitud de comparación fallará si en la solicitud se especifica un objeto desconocido y se devolverá un código de error **noSuchObject**.

La solicitud de comparación fallará si el solicitante de acceso no tiene el permiso **read** para el objeto de acuerdo con el privilegio asignado al solicitante de acceso. Si no se concede el permiso **read** para el objeto, se devolverá un código de error **insufficientAccessRigth** en caso de que el solicitante de acceso tenga el permiso **discloseOnError** para el objeto. En caso contrario, se devolverá un código de error **noSuchObject**.

La solicitud de comparación fallará si el solicitante de acceso no tiene el permiso **compare** para el tipo de atributo en cuestión. Si el solicitante de acceso tiene el permiso **discloseOnError**, se devolverá un código de error **insufficientAccessRigth**. En caso contrario, se devolverá un código de error **noInformation**.

```
compareResult CONTENT-TYPE ::= {
              CompareResult
IDENTIFIED BY id-compareResult
```

El verificador de privilegios utilizará una instancia del tipo de contenido **compareResult** para devolver la información solicitada o comunicar un error.

```
CompareResult ::= SEQUENCE {
  object    DistinguishedName,
  result    CHOICE {
    success   [0] CompareOK,
    failure   [1] AccessdErr,
    ... },
  ... }

CompareOK ::= SEQUENCE {
  matched        [0] BOOLEAN,
  matchedSubtype [1] BOOLEAN OPTIONAL,
  ... }
```

El tipo de datos **CompareResult** especifica la sintaxis del contenido real y tiene los siguientes componentes:

a) el componente **object** contiene el nombre distinguido del objeto con el que se solicita hacer la comparación;

b) el componente resultado contiene el resultado de la solicitud de acceso y tiene dos alternativas:

   – si se selecciona la alternativa **success**, se devolverá una instancia del tipo de datos **CompareOK** con los siguientes componentes:

     i) sl componente **matched** tendrá el valor **TRUE** si un atributo de tipo de atributo o uno de sus subtipos tiene un valor igual al de la solicitud. Además, el componente **matchedSubtype** estará presente y tendrá el valor **TRUE** si los subtipos coinciden. El componente **matched** tendrá el valor **FALSE** si no hay coincidencias para el tipo de atributo o uno de sus subtipos;

   – se escoge la alternativa **failure** si se ha de devolver un error.

8.6 Operación añadir

La operación añadir está compuesta por una solicitud de adición y el correspondiente resultado de adición.

Las solicitudes de adición se transportan en una instancia del tipo de contenido **addRequest** y los resultados de adición se transportan en una instancia del tipo de contenido **addResult**.

```
addRequest CONTENT-TYPE ::= {
              AddRequest
IDENTIFIED BY id-addRequest }
```

Se utiliza una instancia del tipo de contenido **addRequest** para añadir un nuevo objeto al sistema de información.

```
AddRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object    [1] DistinguishedName,
  attr      [2] SEQUENCE SIZE (1..MAX) OF Attribute {{SupportedAttributes}}
                OPTIONAL,
  ... }
```

El tipo de datos **AddRequest** especifica la sintaxis del contenido real y tiene los siguientes componentes:

a) el componente **object** contiene el nombre distinguido del nuevo objeto que se ha de añadir;

b) el componente **attr**, de estar presente, contiene uno o más atributos que se han de asociar con el nuevo objeto.

Si el solicitante de acceso no tiene el permiso **add** para la clase de objetos, se devolverá un código de error **insufficientAccessRigth**.

Si ya existe un objeto con el nombre distinguido facilitado y el solicitante de acceso dispone del permiso **discloseOnError**, se devolverá un código de error **objectAlreadyExists**. En caso contrario se devolverá un código de error **insufficientAccessRigth**.

Si el solicitante de acceso no tiene el permiso **add** para todos los atributos que se han de incluir con el objeto, la solicitud fallará. Si el solicitante de acceso tiene el permiso **discloseOnError** para todos los tipos de atributos enumerados, se devolverá un código de error **insufficientAccessRigth**. En caso contrario, se devolverá un código de error **noInformation**.

```
addResult CONTENT-TYPE ::= {
              AddResult
IDENTIFIED BY id-addResult }
```

El verificador de privilegios utilizará una instancia del tipo de contenido **addResult** para devolver la información solicitada o comunicar un error.

```
AddResult ::= CHOICE {
  success   [0] NULL,
  failure   [1] AccessdErr,
  ... }
```

El tipo de datos **AddResult** especifica la sintaxis del contenido real y tiene los siguientes componentes:

a) la alternativa **success**, que se escoge si se ha añadido el objeto;

b) la alternativa **failure**, que se escoge si se devuelve un error.

8.7 Operación suprimir

La operación suprimir está compuesta por una solicitud de supresión y el correspondiente resultado de supresión.

Las solicitudes de supresión se transportan en una instancia del tipo de contenido **deleteRequest** y el resultado de supresión se transporta en una instancia del tipo de contenido **deleteResult**.

```
deleteRequest CONTENT-TYPE ::= {
              DeleteRequest
IDENTIFIED BY id-deleteRequest }
```

Se utiliza una instancia del tipo de contenido **deleteRequest** para suprimir un objeto existente del sistema de información.

```
DeleteRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object        DistinguishedName,
  ... }
```

El tipo de datos **DeleteRequest** especifica la sintaxis del contenido real y tiene los siguientes componentes:

a) el componente **object** contiene el nombre distinguido de la entrada que se ha de suprimir.

Si el objeto que se ha de suprimir no existe, se devolverá un código de error **noSuchObject**.

Si el solicitante de acceso no tiene el permiso **delete** para la clase de objetos, se devolverá un código de error **insufficientAccessRigth** en caso de que el solicitante de acceso tenga el permiso **discloseOnError** para el objeto. En caso contrario, se devolverá un código de error **noSuchObject**.

```
deleteResult CONTENT-TYPE ::= {
             DeleteResult
IDENTIFIED BY id-deleteResult }
```

El verificador de privilegios utilizará una instancia del tipo de contenido **deleteResult** para devolver la información solicitada o comunicar un error.

```
DeleteResult ::= CHOICE {
  success   [0] NULL,
  failure   [1] AccessdErr,
  ... }
```

El tipo de datos **DeleteResult** tiene dos alternativas:

a) la alternativa **success**, que se escoge si se ha realizado la supresión;

b) la alternativa **failure**, que se escoge si se ha de comunicar un error.

8.8 Operación modificar

La operación modificar está compuesta por una solicitud de modificación y el correspondiente resultado de modificación.

Las solicitudes de modificación se transportan en una instancia del tipo de contenido **modifyRequest** y el resultado de modificación se transporta en una instancia del tipo de contenido **modifyResult**.

```
modifyRequest CONTENT-TYPE ::= {
              ModifyRequest
IDENTIFIED BY id-modifyRequest }
```

Se utiliza una instancia del tipo de contenido **modifyRequest** para modificar un objeto existente.

```
ModifyRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object        DistinguishedName,
  changes       SEQUENCE SIZE (1..MAX) OF ObjectModification,
  select        InformationSelection,
  ... }
```

```
ObjectModification ::= CHOICE {
  addAttribute     [0]  Attribute{{SupportedAttributes}},
  deleteAttribute  [1]  AttributeType,
  addValues        [2]  Attribute{{SupportedAttributes}},
  deleteValues     [3]  Attribute{{SupportedAttributes}},
  replaceAttribute [4]  Attribute{{SupportedAttributes}},
  ... }
```

El tipo de datos **ModifyRequest** especifica la sintaxis del contenido real y tiene los siguientes componentes:

a) El componente **object** contiene el nombre distinguido del objeto que se ha de modificar:

- si el objeto no existe, se devolverá un código de error **noSuchObject**;
- si el solicitante de acceso no tiene el permiso **modify** para ese objeto, se devolverá un código de error **insufficientAccessRigth** en caso de que el solicitante de acceso tenga el permiso **discloseOnError** para el objeto. En caso contrario, se devolverá un código de error **noSuchObject**.

b) El componente **changes** contiene información para modificar uno o más atributos:

- La alternativa **addAttribute** contiene el nuevo atributo que se ha de añadir.
  - i) si el solicitante de acceso no tiene el permiso **add** para el tipo de atributo, se devolverá un código de error **insufficientAccessRigth**;
  - ii) si ya existe un atributo de ese tipo, se devolverá un código de error **attributeAlreadyExists** en caso de que el solicitante de acceso tenga el permiso **discloseOnError** para ese tipo de atributo. En caso contrario, se devolverá un código de error **insufficientAccessRigth**.
- La alternativa **deleteAttribute** identifica el atributo que se ha de suprimir;
  - i) si el solicitante de acceso no tiene el permiso **delete** para el tipo de atributo, se devolverá un código de error **insufficientAccessRigth**;
  - ii) si no existe un atributo de ese tipo, se devolverá un código de error **noSuchAttribute**.
- La alternativa **addValues** identifica un atributo existente por el tipo de atributo. Los valores que se ha de añadir al atributo son los valores incluidos en esta alternativa.
  - i) si el objeto no tiene atributos de ese tipo concreto, se devolverá un código de error **noSuchAttribute** en caso de que el solicitante de acceso tenga el permiso **discloseOnError**. En caso contrario, se devolverá un código de error **insufficientAccessRigth**;
  - ii) si el solicitante de acceso no tiene el permiso **addValue**, se devolverá un código de error **insufficientAccessRigth**;
  - iii) si se intenta añadir un valor ya existente, se devolverá un código de error **attributeValueAlreadyExists** en caso de que el solicitante de acceso tenga el permiso **discloseOnError**. En caso contrario, se devolverá un código de error **insufficientAccessRigth**.
- La alternativa **deleteValues** identifica un atributo por el tipo de atributo. Los valores que se han de suprimir del atributo son los valores incluidos en esta alternativa.
  - i) si el objeto no tiene atributos de ese tipo concreto, se devolverá un código de error **noSuchAttribute** en caso de que el solicitante de acceso disponga del permiso **discloseOnError**. En caso contrario, se devolverá un código de error **insufficientAccessRigth**;
  - ii) si el solicitante de acceso no tiene el permiso **deleteValue**, se devolverá un código de error **insufficientAccessRigth** en caso de que el solicitante de acceso tenga el

permiso **discloseOnError**. En caso contrario, se devolverá un código de error **noSuchAttributeValue**;

iii) si el solicitante de acceso intenta suprimir un valor de atributo que no existe, se devolverá un código de error **noSuchAttributeValue**.

– La alternativa **replaceAttribute** sustituye un atributo existente por un nuevo atributo del mismo tipo:

i) si el objeto no tiene atributos de ese tipo concreto, se devolverá un código de error **noSuchAttribute** en caso de que el solicitante de acceso tenga el permiso **discloseOnError**. En caso contrario, se devolverá un código de error **insufficientAccessRigth**;

ii) si el solicitante de acceso no tiene el permiso **replaceAttribute**, se devolverá un código de error **insufficientAccessRigth** en caso de que el solicitante de acceso tenga el permiso **discloseOnError**. En caso contrario, se devolverá un código de error **noSuchAttribute**.

```
modifyResult CONTENT-TYPE ::= {
              ModifyResult
IDENTIFIED BY id-modifyResult }
```

El verificador de privilegios utilizará una instancia del tipo de contenido **modifyResult** para devolver la información solicitada o comunicar un error.

```
ModifyResult ::= SEQUENCE {
  result    CHOICE {
    success   [0] ObjectInformation,
    failure   [1] AccessdErr,
    ... },
  ... }
```

El tipo de datos **ModifyResult** tiene dos alternativas:

a) La alternativa **success** se escoge si se ha realizado la modificación del objetol

b) La alternativa **failure** se escoge si se ha de devolver un error.

8.9 Operación renombrar objeto

La operación renombrar objeto está compuesta por una solicitud de redenominación y el correspondiente resultado de redenominación.

Las solicitudes de redenominación se transportan en una instancia del tipo de contenido **renameRequest** y el resultado de redenominación se transporta en una instancia del tipo de contenido **renameResult**.

```
renameRequest CONTENT-TYPE ::= {
              RenameRequest
IDENTIFIED BY id-renameRequest }
```

Se utiliza una instancia del tipo de contenido **renameRequest** para cambiar el nombre de un objeto existente.

```
RenameRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object        DistinguishedName,
  new           DistinguishedName,
  ... }
```

El tipo de datos **RenameRequest** especifica la sintaxis del contenido real y tiene los siguientes componentes:

a) El componente **object** contiene el nombre distinguido actual del objeto que se ha de renombrar.

b) El componente **new** contiene el nuevo nombre distinguido del objeto.

Si el objeto que se ha de renombrar no existe, se devolverá un código de error **noSuchObject**.

Si el solicitante de acceso no tiene el permiso **rename** para el objeto nombrado, se devolverá un código de error **insufficientAccessRigth** en caso de que el solicitante de acceso tenga el permiso **discloseOnError** para el objeto nombrado. En caso contrario, se devolverá un código de error **noSuchObject**.

```
renameResult CONTENT-TYPE ::= {
              RenameResult
IDENTIFIED BY id-renameResult }

RenameResult ::= SEQUENCE {
  result    CHOICE {
    success   [0] NULL,
    failure   [1] AccessdErr,
    ... },
  ... }
```

El tipo de datos **RenameResult** tiene dos alternativas:

a) La alternativa **success** se escoge si se ha realizado la modificación del objeto.

b) La alternativa **failure** se escoge si se ha de devolver un error.

8.10 Tratamiento de errores

Cuando se da una condición de excepción durante el tratamiento de una solicitud, el receptor devolverá un código de error incluyendo una instancia del tipo de datos **AccessErr** en el resultado.

```
AccessdErr ::= CHOICE {
  cmsErr     [0] CmsErr,
  ActErr   [1] PbactErr,
  ... }
```

La alternativa **cmsErr** se escogerá si la condición de excepción ocurre durante la evaluación de los tipos de contenido definidos por CMS (véase la cláusula A.5).

La alternativa **pbactErr** se escogerá si no se detectan errores al evaluar las instancias de los tipos de contenido CMS, pero se detecta un error en una instancia encapsulada de un tipo de contenido PBACT específico.

8.11 Selección de información

Se utiliza el tipo de datos **InformationSelection** para especificar qué información se solicita en una solicitud de lectura o de modificación.

```
InformationSelection ::= SEQUENCE {
  attributes      CHOICE {
    allAttributes   [0]  NULL,
    select          [1]  SEQUENCE SIZE (1..MAX) OF ATTRIBUTE.&id,
    ... },
  infoTypes        ENUMERATED {
    attributeTypesOnly     (0),
    attributeTypeAndValues (1),
    ... },
  ...}
```

El tipo de datos **InformationSelection** tiene los siguientes componentes:

a) El componente **attributes** especifica qué atributos se han de devolver. Tiene dos alternativas:
   - La alternativa **allAttributes** se escoge si el solicitante de acceso quiere toda la información sobre un objeto.
   - La alternativa **select** se escoge cuando sólo se solicita un conjunto seleccionado de atributos.

b) El componente **infoTypes** tiene los siguientes elementos de enumeración:
   - El elemento de enumeración **attributeTypesOnly** se escogerá si sólo se han de devolver tipos de atributo. En este caso, el solicitante de acceso deberá tener el permiso **read** para el tipo de atributo de acuerdo con el privilegio vigente. En caso contrario, se elimina el tipo de atributo del resultado. Si el resultado es que no se devuelve información alguna, la solicitud falla.
   - El elemento de enumeración **attributeTypesAndValues** se escogerá si se han de devolver tanto el tipo como los valores de acuerdo con el privilegio vigente. En este caso, el solicitante de acceso deberá tener el permiso **read** para el tipo de atributo de acuerdo con el privilegio vigente. En caso contrario, se suprimen del resultado el tipo de atributo y los valores. Si el resultado es que no se devuelve información alguna, la solicitud falla.

8.12 Información de objeto

Cuando se ha de devolver información sobre un objeto, ésta se devolverá en una instancia del siguiente tipo de datos:

```
ObjectInformation ::= SEQUENCE {
  object  DistinguishedName,
  info    CHOICE {
    attr    SET SIZE (1..MAX) OF Attribute {{SupportedAttributes}},
    type    SET SIZE (1..MAX) OF AttributeType },
  ... }
```

El componente **object** contiene el nombre distinguido del objeto cuya información se devuelve.

El componente **info** contiene una serie de atributos con la información solicitada o un conjunto de tipos de atributo.

Si no hay información que devolver, la solicitud falla.

8.13 Códigos de error definidos

Los códigos de error para los tipos de contenido PBACT específicos se definen a continuación.

```
PbactErr ::= ENUMERATED {
  noSuchService,
  invalidOperationForService,
  insufficientAccessRigth,
  noSuchObject,
  noSuchAttribute,
  noSuchAttributeValue,
  objectAlreadyExists,
  attributeAlreadyExists,
  attributeValueAlreadyExists,
  noInformation,
  ... }
```

a) Se devolverá el código de error **noSuchService** si el solicitante de acceso especifica un servicio para el que no tiene permiso y no está autorizado a saber qué servicio o por qué no se soporta.

b) Se devolverá el código de error **invalidOperationForService** si una operación solicitada no es pertinente para el servicio solicitado.

c) Se devolverá el código de error **insufficientAccessRigth** cuando el solicitante de acceso solicite un servicio para el que no tiene permiso o cuando quiera realizar una operación para la que no tiene permiso de acuerdo con el servicio al que se ha de acceder.

d) Se devolverá el código de error **noSuchObject** si el solicitante de acceso intenta acceder a un objeto inexistente o a un objeto cuya existencia no tiene permiso para conocer.

e) Se devolverá el código de error **noSuchAttribute** si el solicitante de acceso intenta acceder a un atributo inexistente o a un atributo cuya existencia no tiene permiso para conocer.

f) Se devolverá el código de error **noSuchAttributeValue** si el solicitante de acceso intenta acceder a un valor de atributo inexistente o a un valor de atributo cuya existencia no tiene permiso para conocer.

g) Se devolverá el código de error **objectAlreadyExists** si se intenta añadir un objeto con un nombre distinguido igual al nombre distinguido de un objeto ya existente, siempre y cuando el solicitante de acceso tenga permiso para conocer la existencia de ese objeto.

h) Se devolverá el código de error **attributeAlreadyExists** si se intenta añadir un atributo de un tipo ya existente al objeto en cuestión, siempre y cuando el solicitante de acceso tenga permiso para conocer la existencia de ese tipo de atributo en el objeto.

i) Se devolverá el código de error **attributeValueAlreadyExists** si se intenta añadir un atributo a un objeto que ya tiene un atributo del mismo tipo, siempre y cuando el solicitante de acceso tenga permiso para conocer la existencia de ese atributo.

j) Se devolverá el código de error **noInformation** si la información solicitada no está disponible o el solicitante de acceso no tiene permiso para conocer la existencia de esos datos.

9 Protocolo de asignación de privilegios

9.1 Alcance del protocolo

El protocolo de asignación de privilegios se utiliza para asignar privilegios:

a) de una SOA a una AA intermedia para su ulterior delegación;

b) de una SOA directamente a una entidad, utilizando los privilegios asignados para aseverar esos privilegios;

c) de una AA directamente a una entidad, utilizando los privilegios asignados para aseverar esos privilegios; y

d) cuando múltiples AA están en el trayecto entre la SOA y el titular de los privilegios, de una AA a otra AA.

NOTA – Se recomienda que el trayecto de delegación sea lo más corto posible.

9.2 Tipos de contenido

El protocolo de asignación de privilegios utiliza dos tipos de contenido: un tipo de contenido para asignar privilegios y un tipo de contenido para confirmar la asignación.

9.2.1 Tipo de contenido solicitud de asignación de privilegios

Las solicitudes de asignación de privilegios se transportan en una instancia del tipo de contenido **privAssignRequest**.

```
privAssignRequest CONTENT-TYPE ::= {
            PrivAssignRequest
IDENTIFIED BY id-privAssignRequest }
```

La sintaxis del contenido real se especifica mediante el siguiente tipo de datos:

```
PrivAssignRequest ::= SEQUENCE {
  attrCerts   AttributeCertificates OPTIONAL,
  ... }
```

Este tipo de datos sólo tiene un componente que contiene una secuencia de certificados de atributo. Si la solicitud de asignación de privilegios se envía desde una SOA, la secuencia consiste de un solo certificado de atributo. Si hay una única AA entre la SOA y el titular de privilegios, la solicitud de la AA al titular de privilegios incluirá tanto el certificado emitido por la SOA como el certificado de atributo emitido por la AA. Se necesitará un certificado de atributo adicional por cada nueva AA entre la SOA y el titular de privilegios.

9.2.2 Tipo de contenido resultado de asignación de privilegios

Los resultados de asignación de privilegios se transportan en una instancia del tipo de contenido **privAssignResult**.

```
privAssignResult CONTENT-TYPE ::= {
              PrivAssignResult
IDENTIFIED BY id-privAssignResult }
```

La sintaxis del contenido real se especifica mediante el siguiente tipo de datos:

```
PrivAssignResult ::= SEQUENCE {
  result  CHOICE {
    success  NULL,
    failure  PrivAssignErr },
  ... }

PrivAssignErr ::= CHOICE {
--cmsErr     [0] CmsErr,
  assignErr  [1] AssignErr,
  ... }
```

9.2.3 Códigos de error definidos

```
AssignErr ::= ENUMERATED {
  invalidAttributeCertificate (0),
  invalidDelegationPath
  invalidPublicKeyCertificate
    ... }
```

Anexo A

Atribución de identificador de objetos para la serie de Recomendaciones UIT-T X.1080

(Este anexo es parte integrante de esta Recomendación.)

A.1 Nivel superior del árbol de identificadores de objeto

En el Anexo A de [UIT-T X.1081] se atribuyen arcos por debajo del arco atribuido a la telebiometría, que es:

```
id-telebio OBJECT IDENFIER ::= { joint-iso-itu-t(2) telebiometrics(42) }
```

Por debajo de este arco, [UIT-T 1081] atribuye el siguiente arco para la telesalud:

```
id-th OBJECT IDENTIFIER ::= { id-telebio th(3) }
```

En [UIT-T X.1080.1] se han atribuido varios arcos por debajo del arco `id-th`. El arco '0' está atribuido a los módulos ASN.1 definidos en [UIT-T 1080.1]. Se han atribuido otros arcos a categorías de entidades. En esta Recomendación se especifica que el arco '0' también se utilizará para la atribución de identificadores de objeto para la serie UIT-T X.1080 en general. Para evitar coincidencias, se utiliza el valor 10 para la atribución de identificadores de objeto para la serie UIT-T X.1080.

```
id-telehelth OBJECT IDENTIFIER ::= { id-th all(0) telehealth(10) }
```

Los siguientes arcos están atribuidos a las distintas partes de la serie UIT-T X.1080:

```
id-x1080-0 OBJECT IDENTIFIER ::= { id-telehelth part0(0) }
id-x1080-1 OBJECT IDENTIFIER ::= { id-telehelth part1(1) }
id-x1080-2 OBJECT IDENTIFIER ::= { id-telehelth part2(2) }
----
```

Un arco para una parte concreta de la serie UIT-T X.1080 se divide de la siguiente manera:

– Los módulos tienen el arco '0'.

– Los tipos de contenido CMS tienen el arco '1'.

– Los tipos de atributo tienen el arco '2'.

Una parte concreta puede atribuir arcos adicionales en función de sus necesidades.

Para esta Recomendación, se atribuye el siguiente arco para los módulos:

```
id-x1080-0-module OBJECT IDENTIFIER ::= { id-x1080-0 module(0) }
```

El siguiente arco está atribuido a los tipos de contenido CMS:

```
id-x1080-0-Cont   OBJECT IDENTIFIER ::= { id-x1080-0 cmsCont(1) }
```

El siguiente arco está atribuido a los tipos de atributo utilizados para asignar privilegios:

```
id- x1080-0-attr   OBJECT IDENTIFIER ::= { id-x1080-0 prAttr(2) }
```

A.2 Identificadores de objeto para tipos de contenido CMS

Los siguientes identificadores de objeto están atribuidos a los tipos de contenido definidos para el protocolo de asignación de privilegios y para el protocolo de evaluación de privilegios:

```
id-privAssignReq  OBJECT IDENTIFIER ::= { id-x1080-0-Cont privAssignRequest(1) }
id-privAssignRes  OBJECT IDENTIFIER ::= { id-x1080-0-Cont privAssignResult(2) }
id-readRequest    OBJECT IDENTIFIER ::= { id-x1080-0-Cont readRequest(3) }
id-readResult     OBJECT IDENTIFIER ::= { id-x1080-0-Cont readResult(4) }
id-compareRequest OBJECT IDENTIFIER ::= { id-x1080-0-Cont compareRequest(5) }
id-compareResult  OBJECT IDENTIFIER ::= { id-x1080-0-Cont compareResult(6) }
id-addRequest     OBJECT IDENTIFIER ::= { id-x1080-0-Cont addRequest(7) }
id-addResult      OBJECT IDENTIFIER ::= { id-x1080-0-Cont addResult(8) }
id-deleteRequest  OBJECT IDENTIFIER ::= { id-x1080-0-Cont deleteRequest(9) }
id-deleteResult   OBJECT IDENTIFIER ::= { id-x1080-0-Cont deleteResult(10) }
id-modifyRequest  OBJECT IDENTIFIER ::= { id-x1080-0-Cont modifyRequest(11) }
id-modifyResult   OBJECT IDENTIFIER ::= { id-x1080-0-Cont modifyResult(12) }
id-renameRequest  OBJECT IDENTIFIER ::= { id-x1080-0-Cont renameRequest(13) }
id-renameResult   OBJECT IDENTIFIER ::= { id-x1080-0-Cont renameResult(14) }
```

A.3 Identificadores de objeto para los tipos de atributo de privilegios

```
id-at-accessSer   OBJECT IDENTIFIER ::= { id-pbactPrivAttr 1 }
```

Anexo B

Perfil de sintaxis de mensaje criptográfico

(Este anexo es parte integrante de esta Recomendación.)

B.1 Aspectos generales

La sintaxis de mensaje criptográfico (CMS) está definida en [IETF RFC 5652] y especifica las capacidades de comunicación que permiten la integridad, autentificación y confidencialidad de los datos. [IETF RFC 5083] aporta especificaciones adicionales. En [IETF RFC 5911] e [IETF RFC 6268] se dan nuevos módulos ASN.1 para CMS. En este Anexo se presenta un perfil de CMS para su utilización en especificaciones telebiométricas mediante la referencia a esas especificaciones.

CMS es una especificación versátil que se puede utilizar en muchos entornos distintos. Este perfil no utiliza todas las capacidades de la CMS, pero incluye los tipos de datos CMS que se utilizan en esta Recomendación y en otras especificaciones telebiométricas, que pueden referirse al presente Anexo al abordar su utilización de la CMS.

No se pretende definir una especificación para una implementación no conforme con las RFC del IETF, sino sólo tratar los aspectos de la CMS pertinentes a las especificaciones telebiométricas. En el Apéndice I se presenta un módulo ASN.1 informal que refleja la utilización biométrica de la CMS.

La CMS define distintos tipos de contenido que se pueden utilizar para diversos fines. Las especificaciones telebiométricas utilizan el tipo de contenido **signedData** para la autentificación y la integridad; utilizan el tipo de contenido **envelopedData** para la encriptación y, por tanto, la confidencialidad; y utilizan el tipo de contenido **ct-authEnvelopedData**. El tipo de contenido **signedData** se emplea cuando se necesita la firma digital, mientras que el tipo de contenido **envelopedData** se utiliza cuando se requiere la confidencialidad. En tal caso, se encapsula una instancia del tipo de contenido **envelopedData** en una instancia del tipo de contenido **signedData**. El tipo de contenido **ct-authEnvelopedData** se emplea cuando múltiples mensajes forman una tarea concreta.

En las especificaciones telebiométricas no se utilizan todos los aspectos de los tipos de contenido mencionados. Por consiguiente, en este Anexo se presenta un perfil para la utilización de la CMS en la telebiometría. Para facilitar la referencia, se incluyen aquí los aspectos pertinentes de la CMS.

Si se requiere la confidencialidad, se encapsula una instancia de uno de los tipos de contenido definidos en las especificaciones telebiométricas en una instancia del tipo de contenido **envelopedData**. En caso contrario, se encapsula en una instancia del tipo de contenido **signedData**. Otra posibilidad consiste en incluir esa instancia de tipo de contenido en una instancia del tipo de contenido **ct-authEnvelopedData.**

De conformidad con [IETF RFC 6268] un tipo de contenido se define utilizando la siguiente clase de objeto de información:

```
CONTENT-TYPE ::= TYPE-IDENTIFIER
```

La clase de objeto de información **CONTENT-TYPE** es equivalente a la clase de objeto de información ASN.1 **TYPE-IDENTIFIER**. Se utiliza un objeto de información **CONTENT-TYPE** para vincular el tipo de contenido identificado por un identificador de objeto a la sintaxis abstracta del contenido.

El siguiente tipo de datos **ContentInfo** ofrece la sintaxis general para un tipo de contenido:

```
ContentInfo ::= SEQUENCE {
  contentType  CONTENT-TYPE.&id ({TelebSupportedcontentTypes}),
  content      CONTENT-TYPE.&Type
          ({TelebSupportedcontentTypes}{@contentType})OPTIONAL,
  ... }
```

```
TelebSupportedcontentTypes CONTENT-TYPE ::=
  { signedData | envelopedData | ct-authEnvelopedData, ...}
```

Los tipos de contenido soportados son el tipo de contenido **signedData**, el tipo de contenido **envelopedData**, el tipo de contenido **ct-authEnvelopedData** y el conjunto de tipos de contenido definido por cada especificación telebiométrica concreta.

La CMS exige que se especifique la versión de CMS para cada tipo de datos a fin de indicar la sintaxis específica utilizada para ese tipo de datos. Se definen las siguientes versiones:

```
CMSVersion ::= INTEGER{ v0(0), v1(1), v2(2), v3(3), v4(4), v5(5) }
```

En [IETF RFC 6268] se define el siguiente tipo de datos parametrizado que se utiliza en todas las especificaciones:

```
Attributes { ATTRIBUTE:AttrList } ::=
             SET SIZE (1..MAX) OF Attribute {{ AttrList }}
```

B.2 Utilización del tipo de contenido signedData

En la cláusula 5 de [IETF RFC 5652] se especifica el siguiente tipo de contenido. Con una notación ligeramente modificada para reflejar su utilización en telebiometría, se especifica de la siguiente manera:

```
signedData CONTENT-TYPE ::= {
                SignedData
  IDENTIFIED BY id-signedData }

SignedData ::= SEQUENCE  {
  version               CMSVersion (v3),
  digestAlgorithms      SET (SIZE (1)) OF AlgorithmIdentifier
                          {{Teleb-Hash-Algorithms}},
  encapContentInfo      EncapsulatedContentInfo,
  certificates     [0]  IMPLICIT SET (SIZE (1..MAX)) OF Certificate OPTIONAL,
  crls             [1]  IMPLICIT RevocationInfoChoices OPTIONAL,
  signerInfos           SignerInfos,
  ... }
```

NOTA 1 – [IETF RFC 6268] utiliza una versión ligeramente modificada del tipo de datos **AlgorithmIdentifier**. Sin embargo, este perfil utiliza el tipo de datos **AlgorithmIdentifier** definido en [UIT-T X.509].

El componente **version** toma el valor **v3**, de conformidad con la cláusula 5.1 de [IETF RFC 5652].

El componente **digestAlgorithms** está formado por un único elemento que especifica un algoritmo de aleatorización del conjunto de algoritmos de aleatorización aplicables, como se define en la especificación telebiométrica aplicable.

La siguiente definición permite la inclusión de cualquier algoritmo de aleatorización.

```
Teleb-Hash-Algorithms ALGORITHM ::= {...}
```

NOTA 2 – Este perfil no exige un algoritmo de aleatorización específico. Las especificaciones de referencia o los acuerdos de implementación podrán sustituir los puntos con un conjunto de algoritmos de aleatorización que se habrán de soportar en un entorno específico.

NOTA 3 – La CMS permite múltiples firmas digitales y, así, múltiples algoritmos de aleatorización. Para las especificaciones telebiométricas no son pertinentes las múltiples firmas digitales.

El componente **encapContentInfo** contiene una instancia del siguiente tipo de datos:

```
EncapsulatedContentInfo ::= SEQUENCE {
  eContentType     CONTENT-TYPE.&id({envelopedData, ...}),
  eContent     [0] EXPLICIT OCTET STRING
```

```
    (CONTAINING CONTENT-TYPE.&Type({envelopedData, ...}
     {@eContentType})) OPTIONAL }
```

Este tipo de datos tiene los siguientes componentes:

a) El componente **eContentType** contiene el identificador de objeto del tipo de contenido encapsulado. Este componente debe contener la identidad del tipo de contenido **envelopedData**, si se requiere encriptación. De no requerirse encriptación, debe contener uno de los tipos de contenido definidos en la especificación telebiométrica pertinente.

b) El componente **econtent** contiene el contenido encapsulado real envuelto en una cadena de bytes. Siempre debe estar presente.

NOTA 4 – Este componente se define como opcional. Sin embargo, todos los tipos de contenido pertinentes tienen un contenido definido.

El componente **certificates** contiene suficientes certificados de clave pública para establecer un único trayecto de certificación, como especifica el tipo de datos **PkiPath**, definido en [UIT-T X.509].

El componente **crls** no es pertinente para las especificaciones telebiométricas y estará ausente.

El componente **signerInfos** contiene una instancia del tipo de datos **SignerInfos**:

```
SignerInfos ::= SET (SIZE (1)) OF SignerInfo

SignerInfo ::= SEQUENCE {
  version                 CMSVersion (v1),
  sid                     SignerIdentifier,
  digestAlgorithm         AlgorithmIdentifier {{Teleb-Hash-Algorithms}},
  signedAttrs        [0]  IMPLICIT Attributes{{SignedAttributes}} OPTIONAL,
  signatureAlgorithm      AlgorithmIdentifier {{Teleb-Signature-Algorithms}},
  signature               SignatureValue,
  unsignedAttrs      [1]  IMPLICIT Attributes {{UnsignedAttributes}} OPTIONAL,
  ... }

SignerIdentifier ::= CHOICE {
  issuerAndSerialNumber    IssuerAndSerialNumber,
  subjectKeyIdentifier [0] SubjectKeyIdentifier,
  ...}

IssuerAndSerialNumber ::= SEQUENCE {
  issuer       Name,
  serialNumber CertificateSerialNumber }

SignedAttributes ATTRIBUTE ::= { contentType | messageDigest, ... }

Teleb-Signature-Algorithms ALGORITHM ::= {...}

SignatureValue ::= OCTET STRING

UnsignedAttributes ATTRIBUTE ::= {...}
```

Este perfil sólo soporta un firmante, por lo que las instancias del tipo de datos **SignerInfos** sólo tendrán un único elemento. El tipo de datos **SignerInfo** tiene los siguientes componentes:

a) El componente **version** especifica la **v1**, de acuerdo con [IETF RFC 5652].

b) El componente **sid** identifica el certificado de clave pública de la entidad extrema del firmante y contiene una instancia del tipo de datos **SignerIdentifier**. Este tipo de datos especifica dos alternativas:

   – la alternativa **issuerAndSerialNumber** identifica el certificado de clave pública de la entidad extrema especificando el nombre distinguido de la CA emisora y el número de serie del certificado de clave pública. Siempre se tomará esta alternativa;

   – no se tomará la alternativa **subjectKeyIdentifier**.

c) El componente **digestAlgorithm** tiene el mismo valor que el utilizado para el componente **digestAlgorithms** del tipo de datos **SignerInfo**.

d) El componente **signedAttrs** contiene una lista de atributos firmados. [IETF RFC 5652] exige que se incluyan, por lo menos, instancias de los tipos de atributo **contentType** y **messageDigest**. Este perfil no exige la inclusión de atributos adicionales, pero las especificaciones de referencia pueden ampliar la lista.

e) El componente **signatureAlgorithm** contiene el algoritmo firma utilizado para crear la firma digital contenida en el componente **signature**.

NOTA 5 – Este perfil no exige un algoritmo de firma concreto. Las especificaciones de referencia o los acuerdos de aplicación podrán sustituir los puntos por un conjunto de algoritmos de firma que deberán soportarse para una especificación telebiométrica concreta.

f) Los componentes **signature** y **unsignedAttrs** especificados en [IETF RFC 5652].

B.3 Utilización del tipo de contenido envelopedData

B.3.1 Aspectos generales

El tipo de contenido **envelopedData** permite la encriptación de los datos. Para ello es necesario establecer claves simétricas compartidas. En [IETF RFC 5652] se presentan distintas técnicas para generar claves simétricas. Para este perfil se necesita el método de acuerdo de claves conocido como Diffie-Hellman (DH). El método de acuerdo de claves DH se especifica en [IETF RFC 2631] para la técnica de curva no elíptica y en [IETF RFC 5753] para las técnicas de curva elíptica.

El método DH da como resultado un secreto compartido que puede utilizarse como material de encriptación para la generación de claves simétricas compartidas. Este perfil reconoce dos modos de funcionamiento del método DH: el modo efímero-estático y el modo estático-estático.

El modo efímero-estático exige que el receptor tenga un certificado de clave pública con una clave pública DH certificada por la CA emisora. Este certificado de clave pública estará a disposición del emisor. El emisor creará un nuevo par de claves DH para cada mensaje que envíe. De este modo, el secreto compartido será distinto para cada mensaje.

El modo estático-estático exige que cada una de las entidades comunicantes tenga un certificado de clave pública DH certificada. Como de este modo el secreto compartido sería el mismo para cada mensaje, el emisor debe facilitar algún tipo de material de encriptación de usuario aleatorio para que cada mensaje tenga una encriptación distinta.

Este perfil exige que se soporte el modo efímero-estático.

Ambos métodos exigen que las dos entidades de una comunicación tengan el certificado de clave pública de entidad extrema DH certificada de su contraparte, pues la comunicación es bidireccional.

El siguiente tipo de contenido está especificado en la cláusula 6 de [IETF RFC 5652], actualizado en [IETF RFC 6268]:

```
envelopedData CONTENT-TYPE ::= {
                EnvelopedData
  IDENTIFIED BY id-envelopedData }

EnvelopedData ::= SEQUENCE {
  version                   CMSVersion(v0 | v2),
  originatorInfo         [0] IMPLICIT OriginatorInfo OPTIONAL,
  recipientInfos            RecipientInfos,
  encryptedContentInfo      EncryptedContentInfo,
  ...,
  [[2: unprotectedAttrs  [1] IMPLICIT Attributes
       {{UnprotectedAttributes}} OPTIONAL ]] }

UnprotectedAttributes ATTRIBUTE ::=
```

```
  { aa-CEKReference | aa-CEKMaxDecrypts | aa-KEKDerivationAlg }
```

El componente **EnvelopedData** tiene los siguientes componentes:

a) El componente **version** toma, de conformidad con [IETF RFC 5652], el valor **v2** si el componente **unprotectedAttrs** está presente. En caso contrario, toma el valor **v0**.

b) El componente **originatorInfo** está ausente.

c) El componente **recipientInfos** contiene una instancia del tipo de datos **RecipientInfos** especificado en la cláusula B.3.2.

d) El componente **encryptedContentInfo** contiene una instancia del tipo de datos **EncryptedContentInfo** especificado en la cláusula B.3.4.

e) El componente **unprotectedAttrs** es necesario si se prevé que lo siguiente en transmitirse en esa dirección es una instancia del tipo de contenido **ct-authEnvelopedData**. En caso contrario, puede estar ausente.

B.3.2 Información del receptor

El tipo de datos **RecipientInfos** permite múltiples instancias del tipo de datos **RecipientInfo**. Sin embargo, a los efectos de este perfil, se limitará a una única instancia. Una instancia del tipo de datos **RecipientInfo** especifica distintas alternativas de cómo establecer un secreto compartido entre dos partes comunicantes.

```
RecipientInfos ::= SET SIZE (1) OF RecipientInfo

RecipientInfo ::= CHOICE {
  ktri      KeyTransRecipientInfo,
  kari  [1] KeyAgreeRecipientInfo,
  kekri [2] KEKRecipientInfo,
  pwri  [3] PasswordRecipientinfo,
  ori   [4] OtherRecipientInfo,
  ... }
```

Este perfil sólo utiliza dos de las alternativas **RecipientInfo** definidas en [IETF RFC 5652]:

a) La alternativa **kari** es la única que se utilizará para el tipo de contenido **envelopedData**. El tipo de datos **KeyAgreeRecipientInfo** ofrece la información necesaria para el establecimiento de un secreto compartido, como se detalla en la cláusula B.3.3;

b) La alternativa **kekri** es la única que se utilizará para el tipo de contenido **ct-authEnvelopedData**. El tipo de datos **KEKRecipientInfo** ofrece la información necesaria para establecer un secreto compartido, como se detalla en la cláusula B.4.

B.3.3 Acuerdo de claves

```
KeyAgreeRecipientInfo ::= SEQUENCE {
  version                CMSVersion (v3),
  originator         [0] EXPLICIT OriginatorIdentifierOrKey,
  ukm                [1] EXPLICIT UserKeyingMaterial OPTIONAL,
  keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
  recipientEncryptedKeys RecipientEncryptedKeys,
  ... }

OriginatorIdentifierOrKey ::= CHOICE {
  issuerAndSerialNumber    IssuerAndSerialNumber,
  subjectKeyIdentifier [0] SubjectKeyIdentifier,
  originatorKey        [1] OriginatorPublicKey,
  ... }

OriginatorPublicKey ::= SEQUENCE {
  algorithm AlgorithmIdentifier {{SupportedDHPublicKeyAlgorithms}},
  publicKey BIT STRING,
```

```
  ... }

SupportedDHPublicKeyAlgorithms ALGORITHM ::= {...}

UserKeyingMaterial ::= OCTET STRING (SIZE (64))

KeyEncryptionAlgorithmIdentifier ::=
    AlgorithmIdentifier{{SupportedKeyIncryptAlgorithms}}

SupportedKeyIncryptAlgorithms ALGORITHM ::= {...}

RecipientEncryptedKeys ::= SEQUENCE (SIZE (1)) OF RecipientEncryptedKey

RecipientEncryptedKey ::= SEQUENCE {
  rid          KeyAgreeRecipientIdentifier,
  encryptedKey EncryptedKey }

KeyAgreeRecipientIdentifier ::= CHOICE {
  issuerAndSerialNumber IssuerAndSerialNumber,
--rKeyId             [0] IMPLICIT RecipientKeyIdentifier,
  ... }

EncryptedKey ::= OCTET STRING
```

El tipo de datos **KeyAgreeRecipientInfo** tiene los siguientes componentes:

a) El componente **version** toma, de conformidad con [IETF RFC 5652], el valor **v3**.

b) El componente **originator** contiene una instancia del tipo de datos **OriginatorIdentifierOrKey** con las siguientes alternativas:

- La alternativa **issuerAndSerialNumber** se toma si se utiliza el método estático-estático. En ese caso, contendrá una instancia del tipo de datos **IssuerAndSerialNumber**. Este tipo de datos identifica el certificado de clave pública DH del emisor.
  - i) El componente **issuer** contiene el nombre distinguido de la CA emisora y es igual al componente **issuer** del certificado de clave pública en cuestión.
  - ii) El componente **serialNumber** es igual al componente **serialNumber** del certificado de clave pública en cuestión.
- La alternativa **subjectKeyIdentifier** no se toma.
- La alternativa **originatorKey** se toma si se utiliza el método DH efímero-estático. En ese caso, contendrá una instancia del tipo de datos **OriginatorPublicKey** con los siguientes componentes:
  - i) El componente **algorithm** contiene una referencia al algoritmo de clave pública DH utilizado.

NOTA 1 – Este perfil no obliga a utilizar un algoritmo de clave pública DH concreto. Las especificaciones de referencia o los acuerdos de aplicación podrán sustituir los puntos por un conjunto de algoritmos de clave pública DH que habrán de soportarse en un entorno específico.

  - ii) El componente **publicKey** contiene la clave pública DH generada por el emisor. El emisor generará un nuevo par de claves DH cada vez que utilice este tipo de contenido.

A partir de la clave privada local y la clave pública del receptor, el emisor puede generar un secreto compartido. El receptor generará un secreto compartido idéntico utilizando su clave privada y la clave pública del emisor, facilitada en el tipo de datos **OriginatorPublicKey** o el tipo de datos **IssuerAndSerialNumber**.

c) El componente **ukm** está presente si se utiliza el método estático-estático y contiene una instancia del tipo de datos **UserKeyingMaterial**.

A partir del secreto compartido, el valor del componente **ukm** (de ser pertinente) y otra información especificada en [IETF RFC 2631], ambas partes generan la denominada clave de encriptación de claves (KEK). Posteriormente, esta clave se utiliza para encriptar la clave de encriptación de contenido (CEK) generada por el emisor. Esta técnica se denomina encriptación de claves.

d) El componente **keyEncryptionAlgorithm** especifica el algoritmo de encriptación de claves y contiene una instancia del tipo de datos **KeyEncryptionAlgorithmIdentifier.**

NOTA 2 – Este perfil no obliga a utilizar un conjunto concreto de algoritmos de encriptación de claves. En el futuro podrán definirse nuevos algoritmos. Los algoritmos de encriptación de claves de la norma de encriptación avanzada (AES) se definen en [IETF RFC 3394]. Las especificaciones de referencia o los acuerdos de aplicación podrán sustituir los puntos por un conjunto de algoritmos de encriptación que habrá de soportarse en un entorno específico.

e) El componente **recipientEncryptedKeys** contiene una instancia del tipo de datos **RecipientEncryptedKeys**. Tal instancia estará formada de un solo elemento, a saber, una única instancia del tipo de datos **RecipientEncryptedKey**. Este tipo de datos tiene los dos componentes siguientes:

- El componente **rid** contiene la identificación del receptor por su certificado de clave pública de entidad extrema.
- El componente **encryptedKey** contiene la CEK encriptada utilizada para encriptar el contenido, como se ha expuesto en el punto c).

B.3.4 Reutilización de las claves de encriptación de contenido en la CMS

Si se ha de utilizar la CEK para un tipo de contenido **ct-authEnvelopedData** posterior, como se especifica en [IETF RFC 3185], se introducirá la información de referencia conveniente en los atributos no protegidos, como se indica en la cláusula B.3.1. Ambas partes retendrán esta información. Si se exige un alto nivel de seguridad, el atributo de tipo **aa-CEKMaxDecrypts** tendrá el valor '1' o se omitirá.

B.3.5 Información de contenido encriptada

Una instancia del tipo de datos **EncryptedContentInfo** contiene el contenido encapsulado encriptado.

```
EncryptedContentInfo ::= SEQUENCE {
  contentType                 CONTENT-TYPE.&id ({EncryptedContentSet}),
  contentEncryptionAlgorithm SEQUENCE {
    algorithm                   ALGORITHM.&id ({SymmetricEncryptionAlgorithms}),
    parameter                   ALGORITHM.&Type
              ({SymmetricEncryptionAlgorithms}{@.algorithm})} OPTIONAL,
  encryptedContent       [0] IMPLICIT EncryptedContent OPTIONAL,
  ... }

EncryptedContentSet CONTENT-TYPE ::= {...}

SymmetricEncryptionAlgorithms ALGORITHM ::= {...}

EncryptedContent ::= OCTET STRING
```

El componente **encryptedContentInfo** del tipo de datos **EnvelopedData** contiene una instancia del tipo de datos **EncryptedContentInfo**:

a) El componente **contentType** contiene el tipo de contenido para el contenido encriptado. La lista de posibles tipos de contenido contiene aquéllos para los que la encriptación es una opción.

b) Los componentes **contentEncryptionAlgorithm** y **encryptedContent** exigidos por [IETF RFC 5652].

B.4 Utilización del tipo de contenido AuthEnvelopedData

B.4.1 Aspectos generales

Como se especifica en [UIT-T X.1080.1], los protocolos para la telebiometría en general consisten en un intercambio de establecimiento para iniciar una sesión, seguido de múltiples intercambios de información y, por último, la terminación de la sesión. En tal entorno, puede no ser necesario crear una nueva clave de encriptación de claves para cada mensaje.

En [IETF RFC 5083] se especifica un tipo de contenido **ct-authEnvelopedData** no incluido en [IETF RFC 5652]. Este tipo de contenido permite utilizar técnicas de encriptación autentificada eficientes. Este perfil utiliza los algoritmos AES-GCM definidos en [IETF RFC 5084] además de la reutilización de la CEK, como se define en [IETF RFC 3185]. Pueden consultarse más detalles al respecto en esa especificación.

El tipo de contenido **ct-authEnvelopedData** se define como:

```
ct-authEnvelopedData CONTENT-TYPE ::= {
                AuthEnvelopedData
  IDENTIFIED BY id-ct-authEnvelopedData }

AuthEnvelopedData ::= SEQUENCE {
  version                   CMSVersion (v0),
  originatorInfo       [0] IMPLICIT OriginatorInfo OPTIONAL,
  recipientInfos            RecipientInfos,
  authEncryptedContentInfo EncryptedContentInfo,
  authAttrs            [1] IMPLICIT Attributes {{AuthAttributes}} OPTIONAL,
  mac                       MessageAuthenticationCode,
  unauthAttrs          [2] IMPLICIT Attributes {{UnauthAttributes}} OPTIONAL }

AuthAttributes ATTRIBUTE ::= {...}

MessageAuthenticationCode ::= OCTET STRING

UnauthAttributes ATTRIBUTE ::=
  { aa-CEKReference | aa-CEKMaxDecrypts | aa-KEKDerivationAlg }
```

El tipo de datos **AuthEnvelopedData** tiene los siguientes componentes:

a) El componente **version** toma, de conformidad con [IETF RFC 5083], el valor **v0**.

b) El componente **originatorInfo** está ausente.

c) El componente **recipientInfos** contiene una instancia del tipo de datos **RecipientInfos**. Este tipo de datos se describe en la cláusula B.3.2. La alternativa **kekri**, además de la alternativa **kari**, es pertinente para este tipo de contenido. Cuando se toma la alternativa **kekri**, ésta contiene una instancia del tipo de datos **KEKRecipientInfo** especificado en la cláusula B.4.2.

d) El componente **authEncryptedContentInfo** contiene una instancia del tipo de datos **EncryptedContentInfo** especificado en la cláusula B.3.5.

e) El componente **authAttrsm**, de estar presente, contiene un conjunto de atributos que han de protegerse por autentificación.

f) El componente **mac** contiene el código de autentificación de mensaje (MAC) generado.

g) El componente **unauthAttrs** contiene atributos del mismo tipo que el especificado en el punto e) de la cláusula B.3.1. Si se sabe que la instancia de tipo de contenido es la última de la sesión en cuestión en esa dirección, este componente estará ausente.

B.4.2 Información del receptor KEK

```
KEKRecipientInfo ::= SEQUENCE {
  version                CMSVersion (v4),
  kekid                  KEKIdentifier,
```

```
  keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
  encryptedKey           EncryptedKey }

KEKIdentifier ::= SEQUENCE {
  keyIdentifier OCTET STRING,
  date          GeneralizedTime OPTIONAL,
  other         OtherKeyAttribute OPTIONAL,
  ... }
```

El tipo de datos **KEKRecipientInfo** tiene los siguientes componentes:

a) El componente **version** toma, de conformidad con [IETF RFC 5652], el valor **v4**.

b) El componente **kekid** contiene una instancia del tipo de datos **KEKIdentifier** con los siguientes componentes:

   – El componente **keyIdentifier** contiene el identificador de la CEK retenida de un intercambio anterior, como se especifica en la cláusula B.3.4.

   – Los demás componentes no son necesarios.

c) El componente **keyEncryptionAlgorithm** contiene el algoritmo de encriptación especificado en el punto d) de la cláusula B.3.3. Se recomienda utilizar el mismo algoritmo de encriptación para todas las instancias de contenido de una sesión telebiométrica en particular.

B.5 Atributos

En [IETF RFC 5652] se definen los siguientes tipos de atributo. Se prevé que las instancias de estos tipos de atributo se incluyan como atributos firmados.

```
contentType ATTRIBUTE ::= {
  WITH SYNTAX             CONTENT-TYPE.&id({envelopedData, ...})
  EQUALITY MATCHING RULE objectIdentifierMatch
  SINGLE VALUE            TRUE
  ID                      id-contentType }

messageDigest ATTRIBUTE ::= {
  WITH SYNTAX             OCTET STRING
  EQUALITY MATCHING RULE octetStringMatch
  SINGLE VALUE            TRUE
  ID                      id-messageDigest }
```

En [IETF RFC 3185] se definen los siguientes tipos de atributo. Se prevé que las instancias de estos tipos de atributo se incluyan como atributos no firmados.

```
aa-CEKReference ATTRIBUTE ::= {
  WITH SYNTAX             CEKReference
  EQUALITY MATCHING RULE octetStringMatch
  SINGLE VALUE            TRUE
  ID                      id-aa-CEKReference }

CEKReference ::= OCTET STRING

aa-CEKMaxDecrypts ATTRIBUTE ::= {
  WITH SYNTAX             CEKMaxDecrypts
  EQUALITY MATCHING RULE integerMatch
  SINGLE VALUE            TRUE
  ID                      id-aa-CEKMaxDecrypts }

CEKMaxDecrypts ::= INTEGER

aa-KEKDerivationAlg ATTRIBUTE ::= {
  WITH SYNTAX             KEKDerivationAlgorithm
  EQUALITY MATCHING RULE integerMatch
  SINGLE VALUE            TRUE
```

```
  ID                     id-aa-KEKDerivationAlg }

KEKDerivationAlgorithm ::= SEQUENCE {
  kekAlg       AlgorithmIdentifier,
  pbkdf2Param  PBKDF2-params }

PBKDF2-params ::= SEQUENCE {
  salt CHOICE {
    specified   OCTET STRING,
--  otherSource AlgorithmIdentifier {{PBKDF2-SaltSources}}
    ... },
  iterationCount INTEGER (1..MAX),
  keyLength      INTEGER (1..MAX) OPTIONAL,
  prf            AlgorithmIdentifier {{PBKDF2-PRFs}},
  ... }

PBKDF2-PRFs ALGORITHM ::= {...}

PBKDF2-params ::= SEQUENCE {
  salt CHOICE {
    specified   OCTET STRING,
    otherSource AlgorithmIdentifier {{PBKDF2-SaltSources}} },
  iterationCount INTEGER (1..MAX),
  keyLength      INTEGER (1..MAX) OPTIONAL,
  prf            AlgorithmIdentifier {{PBKDF2-PRFs}} DEFAULT algid-hmacWithSHA1
}


id-pkcs OBJECT IDENTIFIER ::=
   { iso(1) member-body(2) usa(840) rsadsi(113549) pkcs(1) }


id-pkcs-9 OBJECT IDENTIFIER ::= { id-pkcs pkcs-9(9) }

id-aa OBJECT IDENTIFIER ::= { id-pkcs-9 smime(16) attributes(2) }

id-contentType         OBJECT IDENTIFIER ::= { id-pkcs-9 3 }
id-messageDigest       OBJECT IDENTIFIER ::= { id-pkcs-9 4 }
id-aa-CEKReference     OBJECT IDENTIFIER ::= { id aa 30 }
id-aa-CEKMaxDecrypts   OBJECT IDENTIFIER ::= { id aa 31 }
id-aa-KEKDerivationAlg OBJECT IDENTIFIER ::= { id aa 32 }
```

B.6 Códigos de error de la sintaxis de mensaje criptográfico

En [b-IETF RFC 7191] se presenta una lista de los códigos de error CMS para todas las utilizaciones posibles de la CMS. A continuación se indican algunos de esos códigos de error pertinentes a la telebiometría. Puede consultarse la descripción de los códigos de error en [b-IETF RFC 7191].

Cuando en [b-IETF RFC 7191] se hace referencia a un certificado, se trata del certificado de clave pública utilizado para los tipos de contenido definidos CMS.

```
CmsErrorCode ::= ENUMERATED {
  decodeFailure                    (1),
  badContentInfo                   (2),
  badSignedData                    (3),
  badEncapContent                  (4),
  badCertificate                   (5),
  badSignerInfo                    (6),
  badSignedAttrs                   (7),
  badUnsignedAttrs                 (8),
  missingContent                   (9),
  noTrustAnchor                   (10),
  notAuthorized                   (11),
```

```
    badDigestAlgorithm               (12),
    badSignatureAlgorithm            (13),
    unsupportedKeySize               (14),
    unsupportedParameters            (15),
    signatureFailure                 (16),
    incorrectTarget                  (23),
    missingSignature                 (29),
    versionNumberMismatch            (31),
    revokedCertificate               (33),
    badEncryptedData                 (62),
    badEnvelopedData                 (63),
    badKeyAgreeRecipientInfo         (66),
    badKEKRecipientInfo              (67),
    badEncryptContent                (68),
    badEncryptAlgorithm              (69),
    missingCiphertext                (70),
    decryptFailure                   (71),
    badMACAlgorithm                  (72),
    badAuthAttrs                     (73),
    badUnauthAttrs                   (74),
    invalidMAC                       (75),
    mismatchedDigestAlg              (76),
    missingCertificate               (77),
    tooManySigners                   (78),
    missingSignedAttributes          (79),
    derEncodingNotUsed               (80),
    invalidAttributeLocation         (82),
    badAttributes                    (85),
    noMatchingRecipientInfo          (91),
    unsupportedKeyWrapAlgorithm      (92),
    badKeyTransRecipientInfo         (93),
    other                           (127) }
```

Anexo C

Especificación formal de los protocolos de aseveración y asignación de privilegios

(Este anexo es parte integrante de esta Recomendación.)

```
Pbact-access { joint-iso-itu-t(2) telebiometrics(42) e-health-protocol(3)
   modules(0) pbact-access(6) version1(1) }
DEFINITIONS IMPLICIT TAGS ::=
BEGIN

-- EXPORTS All

IMPORTS

  -- from Rec. ITU-T X.501 | ISO/IEC 9594-2

  ATTRIBUTE, Attribute{}, AttributeType, AttributeTypeAndValue,
AttributeValueAssertion, DistinguishedName, OBJECT-CLASS, SupportedAttributes
    FROM InformationFramework {joint-iso-itu-t ds(5) module(1)
informationFramework(1) 8}

  -- from Rec. ITU-T X.509 | ISO/IEC 9594-8

  AttributeCertificate
    FROM AttributeCertificateDefinitions {joint-iso-itu-t ds(5) module(1)
         attributeCertificateDefinitions(32) 8}

  CmsErrorCode, CONTENT-TYPE
    FROM CmsTelebiometric { joint-iso-itu-t(2) telebiometrics(42) th(3) part0(0)
modules(0) cmsProfile(1) version1(1) } ;

accessService ATTRIBUTE ::= {
  WITH SYNTAX  AccessService
  ID           id-at-accessService }

AccessService ::= SEQUENCE {
  serviceId          OBJECT IDENTIFIER,
  objectDef          SEQUENCE SIZE (1..MAX) OF ObjectSel,
  ... }

ObjectSel ::= SEQUENCE {
  objecClass         OBJECT-CLASS.&id,
  objSelect          CHOICE {
    allObj        [0]  TargetSelect,
    objectNames   [1]  SEQUENCE SIZE (1..MAX) OF SEQUENCE {
      object             CHOICE {
        names         [1]  SEQUENCE SIZE (1..MAX) OF DistinguishedName,
        subtree       [2]  DistinguishedName,
        ... },
      select           TargetSelect,
      ... },
    ... },
  ... }

TargetSelect ::= SEQUENCE {
  objOper   ObjectOperations OPTIONAL,
  attrSel   AttributeSel     OPTIONAL,
  ... }
  (WITH COMPONENTS {..., objOper  PRESENT } |
   WITH COMPONENTS {..., attrSel  PRESENT } )
```

```
AttributeSel ::= SEQUENCE {
  attSelect           CHOICE {
    allAttr        [0]  SEQUENCE {
      attrOper1      [0]  AttributeOperations OPTIONAL,
      ... },
    attributes     [1]  SEQUENCE SIZE (1..MAX) OF SEQUENCE {
      select              SEQUENCE SIZE (1..MAX) OF ATTRIBUTE.&id,
      attrOper2      [0]  AttributeOperations OPTIONAL,
      ... },
    ... },
  ... }

ObjectOperations ::= BIT STRING {
  read            (0),
  add             (1),
  modify          (2),
  delete          (3),
  rename          (4),
  discloseOnError (5) }

AttributeOperations ::= BIT STRING {
  read             (0),
  compare          (1),
  add              (2),
  modify           (3),
  delete           (4),
  deleteValue      (5),
  replaceAttribute (6),
  discloseOnError  (7) }

PbactContentTypes CONTENT-TYPE ::= {
    privAssignRequest |
    privAssignResult |
    readRequest |
    readResult |
    compareRequest |
    compareResult |
    addRequest |
    addResult |
    deleteRequest |
    deleteResult |
    modifyRequest |
    modifyResult |
    renameRequest |
    renameResult,
    ... }

CommonReqComp ::= SEQUENCE {
  attrCerts [31]  AttributeCertificates OPTIONAL,
  serviceId [30]  OBJECT IDENTIFIER,
  invokId   [29]  INTEGER,
  ... }

AttributeCertificates ::= SEQUENCE SIZE (1..MAX) OF AttributeCertificate

readRequest CONTENT-TYPE ::= {
              ReadRequest
IDENTIFIED BY id-readRequest }

ReadRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object    [1] DistinguishedName,
```

```
  selection [2] InformationSelection,
  ... }

readResult CONTENT-TYPE ::= {
              ReadResult
IDENTIFIED BY id-readResult }


ReadResult ::= SEQUENCE {
  object    DistinguishedName,
  result    CHOICE {
    success   [0] ObjectInformation,
    failure   [1] AccessdErr,
    ... },
  ... }

compareRequest CONTENT-TYPE ::= {
              CompareRequest
IDENTIFIED BY id-compareRequest }

CompareRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object    [1] DistinguishedName,
  purported [2] AttributeValueAssertion,
  ... }

compareResult CONTENT-TYPE ::= {
              CompareResult
IDENTIFIED BY id-compareResult }

CompareResult ::= SEQUENCE {
  object    DistinguishedName,
  result    CHOICE {
    success   [0] CompareOK,
    failure   [1] AccessdErr,
    ... },
  ... }

CompareOK ::= SEQUENCE {
  matched        [0] BOOLEAN,
  matchedSubtype [1] BOOLEAN DEFAULT FALSE,
  ... }

addRequest CONTENT-TYPE ::= {
              AddRequest
IDENTIFIED BY id-addRequest }

AddRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object    [1] DistinguishedName,
  attr      [2] SEQUENCE SIZE (1..MAX) OF Attribute {{SupportedAttributes}}
                OPTIONAL,
  ... }

addResult CONTENT-TYPE ::= {
              AddResult
IDENTIFIED BY id-addResult }

AddResult ::= CHOICE {
  success   [0] NULL,
  failure   [1] AccessdErr,
  ... }

deleteRequest CONTENT-TYPE ::= {
```

```
              DeleteRequest
IDENTIFIED BY id-deleteRequest }

DeleteRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object        DistinguishedName,
  ... }

deleteResult CONTENT-TYPE ::= {
              DeleteResult
IDENTIFIED BY id-deleteResult }

DeleteResult ::= CHOICE {
  success   [0] NULL,
  failure   [1] AccessdErr,
  ... }

modifyRequest CONTENT-TYPE ::= {
              ModifyRequest
IDENTIFIED BY id-modifyRequest }

ModifyRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object        DistinguishedName,
  changes       SEQUENCE SIZE (1..MAX) OF ObjectModification,
  select        InformationSelection,
  ... }

ObjectModification ::= CHOICE {
  addAttribute     [0]  Attribute{{SupportedAttributes}},
  deleteAttribute  [1]  AttributeType,
  addValues        [2]  Attribute{{SupportedAttributes}},
  deleteValues     [3]  Attribute{{SupportedAttributes}},
  replaceAttribute [4]  Attribute{{SupportedAttributes}},
  ... }

modifyResult CONTENT-TYPE ::= {
              ModifyResult
IDENTIFIED BY id-modifyResult }

ModifyResult ::= SEQUENCE {
  result    CHOICE {
    success   [0] ObjectInformation,
    failure   [1] AccessdErr,
    ... },
  ... }

renameRequest CONTENT-TYPE ::= {
              RenameRequest
IDENTIFIED BY id-renameRequest }

RenameRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object        DistinguishedName,
  new           DistinguishedName,
  ... }

renameResult CONTENT-TYPE ::= {
              RenameResult
IDENTIFIED BY id-renameResult }

RenameResult ::= SEQUENCE {
  result    CHOICE {
    success   [0] NULL,
```

```
    failure   [1] AccessdErr,
    ... },
  ... }

AccessdErr ::= CHOICE {
  cmsErr     [0] CmsErrorCode,
  pbactErr   [1] PbactErr,
  ... }

InformationSelection ::= SEQUENCE {
  attributes      CHOICE {
    allAttributes   [0]  NULL,
    select          [1]  SEQUENCE SIZE (1..MAX) OF ATTRIBUTE.&id,
    ... },
  infoTypes        ENUMERATED {
    attributeTypesOnly     (0),
    attributeTypeAndValue  (1),
    ... },
  ... }

ObjectInformation ::= SEQUENCE {
  name   DistinguishedName,
  info   SET SIZE (1..MAX) OF Attribute {{SupportedAttributes}},
  ... }

PbactErr ::= ENUMERATED {
  noSuchService,
  invalidOperationForService,
  insufficientAccessRigth,
  noSuchObject,
  noSuchAttribute,
  noSuchAttributeValue,
  objectAlreadyExists,
  attributeAlreadyExists,
  attributeValueAlreadyExists,
  noInformation,
  ... }

privAssignRequest CONTENT-TYPE ::= {
             PrivAssignRequest
IDENTIFIED BY id-privAssignRequest }

PrivAssignRequest ::= SEQUENCE {
  attrCerts [1]  AttributeCertificates OPTIONAL,
  ... }

privAssignResult CONTENT-TYPE ::= {
             PrivAssignResult
IDENTIFIED BY id-privAssignResult }

PrivAssignResult ::= SEQUENCE {
  result  CHOICE {
    success  NULL,
    failure  PrivAssignErr },
  ... }

PrivAssignErr ::= CHOICE {
  cmsErr     [0] CmsErrorCode,
  assignErr  [1] AssignErr,
  ... }

AssignErr ::= ENUMERATED {
  invalidAttributeCertificate (0),
  ... }
```

```
-- object identifier allocations

-- top tree

id-pbact             OBJECT IDENTIFIER ::=
  { joint-iso-itu-t(2) telebiometrics(42) e-health-protocol(3) pbact(20) }
id-pbactmodule       OBJECT IDENTIFIER ::= { id-pbact module(0) }
id-pbactCont         OBJECT IDENTIFIER ::= { id-pbact cmsCont(1) }
id-pbactPrivAttr     OBJECT IDENTIFIER ::= { id-pbact prAttr(2) }

-- Content types

id-privAssignRequest OBJECT IDENTIFIER ::= { id-pbactCont privAssignRequest(1) }
id-privAssignResult  OBJECT IDENTIFIER ::= { id-pbactCont privAssignResult(2) }
id-readRequest       OBJECT IDENTIFIER ::= { id-pbactCont readRequest(3) }
id-readResult        OBJECT IDENTIFIER ::= { id-pbactCont readResult(4) }
id-compareRequest    OBJECT IDENTIFIER ::= { id-pbactCont compareRequest(5) }
id-compareResult     OBJECT IDENTIFIER ::= { id-pbactCont compareResult(6) }
id-addRequest        OBJECT IDENTIFIER ::= { id-pbactCont addRequest(7) }
id-addResult         OBJECT IDENTIFIER ::= { id-pbactCont addResult(8) }
id-deleteRequest     OBJECT IDENTIFIER ::= { id-pbactCont deleteRequest(9) }
id-deleteResult      OBJECT IDENTIFIER ::= { id-pbactCont deleteResult(10) }
id-modifyRequest     OBJECT IDENTIFIER ::= { id-pbactCont modifyRequest(11) }
id-modifyResult      OBJECT IDENTIFIER ::= { id-pbactCont modifyResult(12) }
id-renameRequest     OBJECT IDENTIFIER ::= { id-pbactCont renameRequest(13) }
id-renameResult      OBJECT IDENTIFIER ::= { id-pbactCont renameResult(14) }


-- Artribute types for carring privilege definitions

id-at-accessService OBJECT IDENTIFIER ::= { id-pbactPrivAttr 1 }

END
```

Apéndice I

Especificación informal del perfil de sintaxis de mensaje criptográfico

(Este apéndice no forma parte integrante de esta Recomendación.)

Una implementación que sólo soporte el módulo **CmsTelebiometric** no es conforme con la especificación de la CMS del IETF y no se considerará como una especificación de implementación. Se presenta aquí con fines informativos y para verificar la conformidad.

```
CmsTelebiometric { joint-iso-itu-t(2) telebiometrics(42) th(3) part0(0)
   modules(0) cmsProfile(1) version1(1) }
DEFINITIONS ::=
BEGIN

-- EXPORTS All

IMPORTS

  -- from Rec. ITU-T X.501 | ISO/IEC 9594-2

      ATTRIBUTE, Attribute{}, DistinguishedName, objectIdentifierMatch
    FROM InformationFramework {joint-iso-itu-t ds(5) module(1)
informationFramework(1) 8}

  -- from Rec. ITU-T X.509 | ISO/IEC 9594-8

  ALGORITHM, AlgorithmIdentifier, Certificate, CertificateSerialNumber
    FROM AuthenticationFramework {joint-iso-itu-t ds(5) module(1)
authenticationFramework(7) 8}

  -- from Rec. ITU-T X.520 | ISO/IEC 9594-6

  integerMatch, octetStringMatch
    FROM SelectedAttributeTypes {joint-iso-itu-t ds(5) module(1)
selectedAttributeTypes(5) 8}  ;

CONTENT-TYPE ::= TYPE-IDENTIFIER

ContentType ::= CONTENT-TYPE.&id

ContentInfo ::= SEQUENCE {
  contentType  CONTENT-TYPE.&id ({TelebSupportedcontentTypes}),
  content      CONTENT-TYPE.&Type
         ({TelebSupportedcontentTypes}{@contentType})OPTIONAL,
  ... }

TelebSupportedcontentTypes CONTENT-TYPE ::=
  { signedData | envelopedData | ct-authEnvelopedData, ...}

CMSVersion ::= INTEGER{ v0(0), v1(1), v2(2), v3(3), v4(4), v5(5) }

Attributes { ATTRIBUTE:AttrList } ::=
             SET SIZE (1..MAX) OF Attribute {{ AttrList }}

signedData CONTENT-TYPE ::= {
                SignedData
  IDENTIFIED BY id-signedData }

SignedData ::= SEQUENCE  {
  version               CMSVersion (v3),
  digestAlgorithms      SET (SIZE (1)) OF AlgorithmIdentifier
```

```
                          {{Teleb-Hash-Algorithms}},
  encapContentInfo      EncapsulatedContentInfo,
  certificates     [0]  IMPLICIT SET (SIZE (1..MAX)) OF Certificate OPTIONAL,
--crls             [1]  IMPLICIT RevocationInfoChoices OPTIONAL,
  signerInfos           SignerInfos,
  ... }

Teleb-Hash-Algorithms ALGORITHM ::= {...}

EncapsulatedContentInfo ::= SEQUENCE {
  eContentType     CONTENT-TYPE.&id({IncludedContent}),
  eContent     [0] EXPLICIT OCTET STRING
    (CONTAINING CONTENT-TYPE.&Type({IncludedContent}
     {@eContentType})) OPTIONAL }

IncludedContent CONTENT-TYPE ::= {envelopedData, ...}

SignerInfos ::= SET (SIZE (1)) OF SignerInfo

SignerInfo ::= SEQUENCE {
  version                  CMSVersion (v1),
  sid                      SignerIdentifier,
  digestAlgorithm          AlgorithmIdentifier {{Teleb-Hash-Algorithms}},
  signedAttrs         [0]  IMPLICIT Attributes{{SignedAttributes}} OPTIONAL,
  signatureAlgorithm       AlgorithmIdentifier {{Teleb-Signature-Algorithms}},
  signature                SignatureValue,
  unsignedAttrs       [1]  IMPLICIT Attributes {{UnsignedAttributes}} OPTIONAL,
  ... }

SignerIdentifier ::= CHOICE {
  issuerAndSerialNumber    IssuerAndSerialNumber,
--subjectKeyIdentifier [0] SubjectKeyIdentifier,
  ...}

IssuerAndSerialNumber ::= SEQUENCE {
  issuer       DistinguishedName,
  serialNumber CertificateSerialNumber }

SignedAttributes ATTRIBUTE ::= { contentType | messageDigest, ... }

Teleb-Signature-Algorithms ALGORITHM ::= {...}

SignatureValue ::= OCTET STRING

UnsignedAttributes ATTRIBUTE ::= {...}

envelopedData CONTENT-TYPE ::= {
                EnvelopedData
  IDENTIFIED BY id-envelopedData }

EnvelopedData ::= SEQUENCE {
  version                    CMSVersion(v0 | v2),
--originatorInfo          [0] IMPLICIT OriginatorInfo OPTIONAL,
  recipientInfos             RecipientInfos,
  encryptedContentInfo       EncryptedContentInfo,
  ...,
  [[2: unprotectedAttrs  [1] IMPLICIT Attributes
       {{UnprotectedAttributes}} OPTIONAL ]] }

RecipientInfos ::= SET SIZE (1) OF RecipientInfo

UnprotectedAttributes ATTRIBUTE ::=
  { aa-CEKReference | aa-CEKMaxDecrypts | aa-KEKDerivationAlg }
```

```
RecipientInfo ::= CHOICE {
--ktri      KeyTransRecipientInfo,
  kari  [1] KeyAgreeRecipientInfo,
  kekri [2] KEKRecipientInfo,
--pwri  [3] PasswordRecipientinfo,
--ori   [4] OtherRecipientInfo,
  ... }

KeyAgreeRecipientInfo ::= SEQUENCE {
  version                CMSVersion (v3),
  originator         [0] EXPLICIT OriginatorIdentifierOrKey,
  ukm                [1] EXPLICIT UserKeyingMaterial OPTIONAL,
  keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
  recipientEncryptedKeys RecipientEncryptedKeys,
  ... }

OriginatorIdentifierOrKey ::= CHOICE {
  issuerAndSerialNumber    IssuerAndSerialNumber,
--subjectKeyIdentifier [0] SubjectKeyIdentifier,
  originatorKey        [1] OriginatorPublicKey,
  ... }

OriginatorPublicKey ::= SEQUENCE {
  algorithm AlgorithmIdentifier {{SupportedDHPublicKeyAlgorithms}},
  publicKey BIT STRING,
  ... }

SupportedDHPublicKeyAlgorithms ALGORITHM ::= {...}

UserKeyingMaterial ::= OCTET STRING (SIZE (64))

KeyEncryptionAlgorithmIdentifier ::=
   AlgorithmIdentifier{{SupportedKeyIncryptAlgorithms}}

SupportedKeyIncryptAlgorithms ALGORITHM ::= {...}

RecipientEncryptedKeys ::= SEQUENCE (SIZE (1)) OF RecipientEncryptedKey

RecipientEncryptedKey ::= SEQUENCE {
  rid          KeyAgreeRecipientIdentifier,
  encryptedKey EncryptedKey }

KeyAgreeRecipientIdentifier ::= CHOICE {
  issuerAndSerialNumber IssuerAndSerialNumber,
--rKeyId            [0] IMPLICIT RecipientKeyIdentifier,
  ... }

EncryptedKey ::= OCTET STRING

EncryptedContentInfo ::= SEQUENCE {
  contentType                CONTENT-TYPE.&id ({EncryptedContentSet}),
  contentEncryptionAlgorithm SEQUENCE {
    algorithm                  ALGORITHM.&id ({SymmetricEncryptionAlgorithms}),
    parameter                  ALGORITHM.&Type
              ({SymmetricEncryptionAlgorithms}{@.algorithm})} OPTIONAL,
  encryptedContent       [0] IMPLICIT EncryptedContent OPTIONAL,
  ... }

EncryptedContentSet CONTENT-TYPE ::= {...}

SymmetricEncryptionAlgorithms ALGORITHM ::= {...}

EncryptedContent ::= OCTET STRING
```

```
ct-authEnvelopedData CONTENT-TYPE ::= {
                AuthEnvelopedData
  IDENTIFIED BY id-ct-authEnvelopedData }

AuthEnvelopedData ::= SEQUENCE {
  version                   CMSVersion (v0),
--originatorInfo       [0] IMPLICIT OriginatorInfo OPTIONAL,
  recipientInfos            RecipientInfos,
  authEncryptedContentInfo EncryptedContentInfo,
  authAttrs             [1] IMPLICIT Attributes {{AuthAttributes}} OPTIONAL,
  mac                       MessageAuthenticationCode,
  unauthAttrs           [2] IMPLICIT Attributes {{UnauthAttributes}} OPTIONAL }

AuthAttributes ATTRIBUTE ::= {...}

MessageAuthenticationCode ::= OCTET STRING

UnauthAttributes ATTRIBUTE ::=
  { aa-CEKReference | aa-CEKMaxDecrypts | aa-KEKDerivationAlg }

KEKRecipientInfo ::= SEQUENCE {
  version                CMSVersion (v4),
  kekid                  KEKIdentifier,
  keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
  encryptedKey           EncryptedKey }

KEKIdentifier ::= SEQUENCE {
  keyIdentifier OCTET STRING,
--date          GeneralizedTime OPTIONAL,
--other         OtherKeyAttribute OPTIONAL,
  ... }

contentType ATTRIBUTE ::= {
  WITH SYNTAX            CONTENT-TYPE.&id({envelopedData, ...})
  EQUALITY MATCHING RULE objectIdentifierMatch
  SINGLE VALUE           TRUE
  ID                     id-contentType }

messageDigest ATTRIBUTE ::= {
  WITH SYNTAX            OCTET STRING
  EQUALITY MATCHING RULE octetStringMatch
  SINGLE VALUE           TRUE
  ID                     id-messageDigest }

aa-CEKReference ATTRIBUTE ::= {
  WITH SYNTAX            CEKReference
  EQUALITY MATCHING RULE octetStringMatch
  SINGLE VALUE           TRUE
  ID                     id-aa-CEKReference }

CEKReference ::= OCTET STRING

aa-CEKMaxDecrypts ATTRIBUTE ::= {
  WITH SYNTAX            CEKMaxDecrypts
  EQUALITY MATCHING RULE integerMatch
  SINGLE VALUE           TRUE
  ID                     id-aa-CEKReference }

CEKMaxDecrypts ::= INTEGER

aa-KEKDerivationAlg ATTRIBUTE ::= {
  WITH SYNTAX            KEKDerivationAlgorithm
  EQUALITY MATCHING RULE integerMatch
  SINGLE VALUE           TRUE
```

```
  ID                      id-aa-KEKDerivationAlg }

KEKDerivationAlgorithm ::= SEQUENCE {
  kekAlg       AlgorithmIdentifier {{SupportedKeyIncryptAlgorithms}},
  pbkdf2Param  PBKDF2-params }

PBKDF2-params ::= SEQUENCE {
  salt CHOICE {
    specified   OCTET STRING,
--  otherSource AlgorithmIdentifier {{PBKDF2-SaltSources}}
    ... },
  iterationCount INTEGER (1..MAX),
  keyLength      INTEGER (1..MAX) OPTIONAL,
  prf            AlgorithmIdentifier {{PBKDF2-PRFs}},
  ... }

PBKDF2-PRFs ALGORITHM ::= {...}

id-pkcs OBJECT IDENTIFIER ::=
   { iso(1) member-body(2) usa(840) rsadsi(113549) pkcs(1) }

id-pkcs-9 OBJECT IDENTIFIER ::= { id-pkcs pkcs-9(9) }

id-ct OBJECT IDENTIFIER ::= { id-pkcs-9 smime(16) ct(1) }
id-aa OBJECT IDENTIFIER ::= { id-pkcs-9 smime(16) attributes(2) }

id-contentType         OBJECT IDENTIFIER ::= { id-pkcs-9 3 }
id-messageDigest       OBJECT IDENTIFIER ::= { id-pkcs-9 4 }
id-aa-CEKReference     OBJECT IDENTIFIER ::= { id-aa 30 }
id-aa-CEKMaxDecrypts   OBJECT IDENTIFIER ::= { id-aa 31 }
id-aa-KEKDerivationAlg OBJECT IDENTIFIER ::= { id-aa 32 }

id-signedData OBJECT IDENTIFIER ::= {iso(1) member-body(2)
us(840)rsadsi(113549) pkcs(1) pkcs7(7) 2}

id-envelopedData OBJECT IDENTIFIER ::= {iso(1) member-body(2) us(840)
rsadsi(113549) pkcs(1) pkcs7(7) 3}

id-ct-authEnvelopedData OBJECT IDENTIFIER ::= { id-ct 23 }

END -- CmsTelebiometric
```

Bibliografía

[b-UIT-T X.841] Recomendación UIT-T X.841 (2000) | ISO/CEI 15816:2002, *Tecnología de la información – Técnicas de seguridad – Objetos de información de seguridad para control de acceso.*

[b-CEI 62351-8] CEI/TS 62351-8:2011, *Power systems management and associated information exchange – Data and communications security – Part 8: Role based access control.*

[b-NIST 800-56A] NIST Special Publication 800-56A, Revision 2 (2013), *Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography.*

[b-NIST 800-162] NIST Special Publication 800-162 (2014), *Guide to Attribute Based Access Control (ABAC) Definition and Considerations.*

[b-IETF RFC 5480] IETF RFC 5480 (2009), *Elliptic Curve Cryptography Subject Public Key Information.*

[b-IETF RFC 7191] IETF RFC 7191 (2014), *Cryptographic Message Syntax (CMS) – Key Package Receipt and Error Content Types.*

SERIES DE RECOMENDACIONES DEL UIT-T

Serie A	Organización del trabajo del UIT-T
Serie D	Principios de tarificación y contabilidad y cuestiones económicas y políticas de las telecomunicaciones/TIC internacionales
Serie E	Explotación general de la red, servicio telefónico, explotación del servicio y factores humanos
Serie F	Servicios de telecomunicación no telefónicos
Serie G	Sistemas y medios de transmisión, sistemas y redes digitales
Serie H	Sistemas audiovisuales y multimedia
Serie I	Red digital de servicios integrados
Serie J	Redes de cable y transmisión de programas radiofónicos y televisivos, y de otras señales multimedia
Serie K	Protección contra las interferencias
Serie L	Medio ambiente y TIC, cambio climático, ciberdesechos, eficiencia energética, construcción, instalación y protección de los cables y demás elementos de planta exterior
Serie M	Gestión de las telecomunicaciones, incluida la RGT y el mantenimiento de redes
Serie N	Mantenimiento: circuitos internacionales para transmisiones radiofónicas y de televisión
Serie O	Especificaciones de los aparatos de medida
Serie P	Calidad de la transmisión telefónica, instalaciones telefónicas y redes de líneas locales
Serie Q	Conmutación y señalización, y mediciones y pruebas asociadas
Serie R	Transmisión telegráfica
Serie S	Equipos terminales para servicios de telegrafía
Serie T	Terminales para servicios de telemática
Serie U	Conmutación telegráfica
Serie V	Comunicación de datos por la red telefónica
Serie X	**Redes de datos, comunicaciones de sistemas abiertos y seguridad**
Serie Y	Infraestructura mundial de la información, aspectos del protocolo Internet, redes de próxima generación, Internet de las cosas y ciudades inteligentes
Serie Z	Lenguajes y aspectos generales de soporte lógico para sistemas de telecomunicación

Impreso en Suiza
Ginebra, 2017