Departamento de Informática

Seguridad en Servicios Web

C. Gutiérrez, E. Fernández-Medina, M. Piattini

Universidad de Castilla-La Mancha, España.

Informe Técnico UCLM DIAB-05-01-2

Enero 2005

Universidad de
Castilla-La Mancha

Esta investigación es parte de los proyectos CALIPO (TIC2003-07804-C05-03) y RETISTIC (TIC2002-12487-E), soportados por la Dirección General de Investigación del Ministerio de Ciencia y Tecnología.

DEPÓSITO LEGAL:

ÍNDICE

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1. Introducción

En este informe se va a realizar un estudio los aspectos de la seguridad que se deben considerar a la hora de diseñar sistemas basados en los principios y las tecnologías definidas en el contexto de los servicios Web. Explicaremos cuáles son las amenazas así como se indicarán posibles soluciones o, en los peores casos, mecanismos para maximizar la reducción del riesgo.

El informe comienza con un repaso de las principales iniciativas, en forma de consorcios u organizaciones, que están implicadas en los servicios Web en general y en el aspecto de su seguridad en particular.

A continuación se explica de manera resumida el problema de la seguridad tal y como es tratado desde la especificación de la arquitectura de referencia de servicios Web elaborado por el consorcio W3C. En este apartado enumeraremos los escenarios de amenaza a los que se puede ver sometido un sistema basado en servicios Web y mencionaremos los requisitos de seguridad señalados por la arquitectura de referencia y que toda realización de la misma debe considerar. Además, se desarrollará un ejemplo sencillo en el que se introducirán los estándares de seguridad relacionados con los servicios Web de forma que se puede adquirir un conocimiento general sobre ellos y su aplicación práctica.

Posteriormente se desarrollarán, con gran profundidad, los principales estándares de seguridad en servicios Web. Estos estándares nos proporcionarán autenticación, autorización, confidencialidad, integridad y, opcionalmente, no repudio.

A continuación se desarrollará el concepto de federación de la confianza entre dominios mediante servicios Web. Veremos las arquitecturas existentes así como las especificaciones más actuales.

2. VISIÓN GENERAL DE LAS ORGANIZACIONES E INICIATIVAS ACTUALES

2. Visión general de organizaciones e iniciativas actuales.

2.1 Consorcio World Wide Web (W3C)

El Consorcio World Wide Web (http://www.w3.org/Consortium/) se fundó en Octubre de 1994 para liderar el WWW a su máximo potencial mediante el desarrollo de protocolos comunes que fomenten su evolución y aseguren su interoperabilidad. El W3C tiene alrededor de 400 organizaciones miembro procedentes de todo el mundo y ha alcanzado un reconocimiento internacional por su contribución al crecimiento de la Web. En Octubre de 1994, Tim Berners-Lee, inventor de la Web, fundó este consorcio en el Laboratorio de Ciencias de la Computación (LCS) ubicado en el Instituto de Tecnología de Massachussets (MIT) en colaboración con la Organización Europea de Investigación Nuclear (CERN), lugar donde la Web tuvo su origen, con soporte de la Agencia de Proyectos de Investigación de Defensa Avanzada (DARPA) y de la Comisión Europea. Para obtener una información más detallada sobre esta iniciativa conjunta y las contribuciones del CERN, INRIA y MIT, se puede consultar la declaración sobre la Iniciativa Conjunta de la World Wide Web.

En Abril de 1995, el Instituto Nacional de Investigación en Informática y Automatismos (INRIA) se convirtió en la primera delegación europea del W3C, seguido por la Universidad de Japón de Keio (*KEIO*) en Asia en 1996. En el 2003, el Consorcio de Investigación Europeo en Informática

y Matemáticas (ERCIM) tomó el control sobre el mando europeo del W3C cedido por INRIA. El W3C persigue también una audiencia internacional a través de sus oficinas ubicadas por todo el mundo.

2.1.1 Misión del W3C

Mediante el fomento de la interoperabilidad y la promoción de un foro de discusión abierto, la W3C se compromete en liderar la evolución técnica de la Web. En tan sólo 7 años, la W3C ha desarrollado más de 50 especificaciones técnicas orientadas a la infraestructura Web. Sin embargo, la Web continúa siendo joven y "hay todavía mucho trabajo por hacer", especialmente a medida que las computadoras, telecomunicaciones y las tecnologías multimedia convergen. Para cubrir las crecientes necesidades de los usuarios y el incremento de la potencia de las máquinas, el W3C ya se encuentra fabricando los cimientos de la próxima generación de la Web. Las tecnologías W3C ayudarán a hacer de la Web una infraestructura robusta, escalable y adaptativa para el mundo de la información.

2.1.2 Metas del W3C

Los objetivos a largo plazo de la W3C para la Web son:

- Acceso Universal. Hacer que la Web sea accesible por todos fomentando las tecnologías que tienen en cuenta las vastas diferencias de culturas, lenguajes, educación, capacidad, recursos materiales, dispositivos de acceso y limitaciones físicas de los usuarios de todos los continentes.

- Web Semántica. Desarrollar un entorno software que permita a cada usuario llevar a cabo el mejor uso posible de los recursos disponibles en la Web.
- Web de Confianza. Guiar el desarrollo de la Web con cuidada consideración en los aspectos comerciales y sociales originados por esta tecnología.

2.1.3 Rol del W3C

Como muchas otras tecnologías de la información, en particular aquellas que deben su éxito al crecimiento de Internet, la Web debe evolucionar a un ritmo sin comparación respecto a otras industrias. El tiempo para convertir una idea brillante en un nuevo producto o servicio y hacerlo disponible en la Web para el mundo entero debe ser mínimo. Al mismo tiempo, la facilidad de retroalimentación desde el cliente de la Web ha hecho posible para los diseñadores afinar sus productos casi continuamente. Con una audiencia de millones emitiendo especificaciones W3C y proporcionando retroalimentación, el W3C concentra su esfuerzo en tres tareas principales:

- Visión. El W3C fomenta y desarrolla su visión del futuro de la WWW. Las contribuciones realizadas desde varios cientos de investigadores e ingenieros procedentes de las Organizaciones Miembro, Equipos de Trabajo del W3C o de la propia comunidad Web permiten que el W3C identifique los requisitos que deben ser satisfechos para conseguir que la Web sea realmente un espacio de información universal.

- Diseño. La W3C diseña tecnologías Web para fabricar esta visión, teniendo en cuenta las tecnologías existentes así como aquellas que aparecerán en un futuro.
- Estandarización. El W3C contribuye con su esfuerzo en la estandarización de las tecnologías Web produciendo especificaciones (llamadas "Recomendaciones") que describen los bloques de construcción de la Web. La W3C crea estas Recomendaciones (y otros informes técnicos) y las pone a su libre disposición para todos aquellos usuarios que deseen leerlas.

2.1.4 Principios de diseño Web

La Web es una aplicación construida en el punto más alto de Internet y, como tal, ha heredado sus principios de diseño fundamentales:

- Interoperabilidad. Las especificaciones para los lenguajes Web y los protocolos deben ser compatibles entre sí y permitir que el hardware y el software utilizado para acceder a la Web trabaje conjuntamente.

- Evolución. La Web debe ser capaz de acomodar futuras tecnologías. Los principios de diseño tomados como referencia como la simplicidad, modularidad, y extensibilidad incrementan las opciones de que la Web funcione (y funcionará) con tecnologías emergentes tales como dispositivos Web móviles o la televisión digital, así como aquellos aún por llegar.

– Descentralización. La descentralización es sin lugar a dudas el principio más nuevo y más difícil de aplicar. Para que la Web "escale" a proporciones mundiales mientras se muestra resistente a errores y caídas, la arquitectura (como Internet) debe limitar o eliminar las dependencias sobre registros centrales.

Estos son los principios que guían el trabajo llevado a cabo por las Actividades del W3C.

2.1.5 Actividades del W3C

El W3C hace la mayor parte de su trabajo guiado por un mandato explícito de sus miembros. Como se describe en el Documento de Proceso, los Miembros revisan las propuestas en un trabajo denominado "Propuestas de Actividad". Cuando existe un consenso entre los Miembros en perseguir este trabajo, el W3C inicia una nueva Actividad.

El W3C organiza las Actividades generalmente en grupos: 'Working Groups' o Grupos de Trabajo, 'Interest Groups' o Grupos de Interés (para un trabajo más general) y 'Coordination Groups' o Grupos de Coordinación (por ejemplo para gestionar la comunicación entre grupos relacionados). Estos grupos, compuestos de representantes de las Organizaciones Miembro, el Equipo y expertos invitados producen el grueso de los resultados de la W3C: informes técnicos, software libre y abierto, y servicios. Estos grupos también aseguran la coordinación con otros estándares y con otras comunidades técnicas. Existen en la

actualidad alrededor de 30 Grupos de Trabajo o 'Working Groups' en el W3C.

Para facilitar la gestión, el Equipo organiza las Actividades W3C y el resto de trabajo bajo cuatro dominios:

- Dominio de Arquitectura. El Dominio de Arquitectura desarrolla las tecnologías subyacentes de la Web.
- Dominio de Interacción. El Dominio de Interacción busca mejorar la interacción del usuario con la Web y facilitar la autoría para beneficiar por igual a los usuarios y proveedores de los contenidos. También trabaja sobre los formatos y los mensajes que presentarán la información a los usuarios con precisión, belleza y el más alto nivel de control.
- Dominio de Tecnología y Sociedad. Este dominio busca el desarrollo de la Web como infraestructura para cumplir aspectos sociales, legales, y políticos.
- Iniciativa de Accesibilidad a la Web (WAI). El W3C tiene como compromiso desarrollar la Web en su más completo potencial lo que incluye un alto grado de usabilidad para gente con discapacidades. La WAI (Web Access Initivative) persigue la accesibilidad a la Web a través de cinco áreas de trabajo principales: tecnologías, guías, herramientas educación, investigación y desarrollo.

Además, la Actividad de Aseguramiento de la Calidad (Quality Assurance (QA)) y la Política de Patentes (Patent Policy) son aplicables a estos cuatro dominios. Se puede obtener más información sobre las Actividades llevadas a cabo dentro de cada Dominio en la página Web del W3C.

2.1.6 Algunas Recomendaciones W3C realizadas hasta la fecha

Guiados por los principios de diseño mencionados, el W3C ha publicado más de cuarenta Recomendaciones desde su nacimiento. Cada Recomendación no sólo se construye sobre la anterior sino que también es diseñada de forma que pueda ser integrada con futuras especificaciones. El W3C está transformando la arquitectura inicial de la Web (esencialmente HTML, URIs y HTTP) en la arquitectura de la Web del mañana, construido sobre una base sólida proporcionada por el XML.

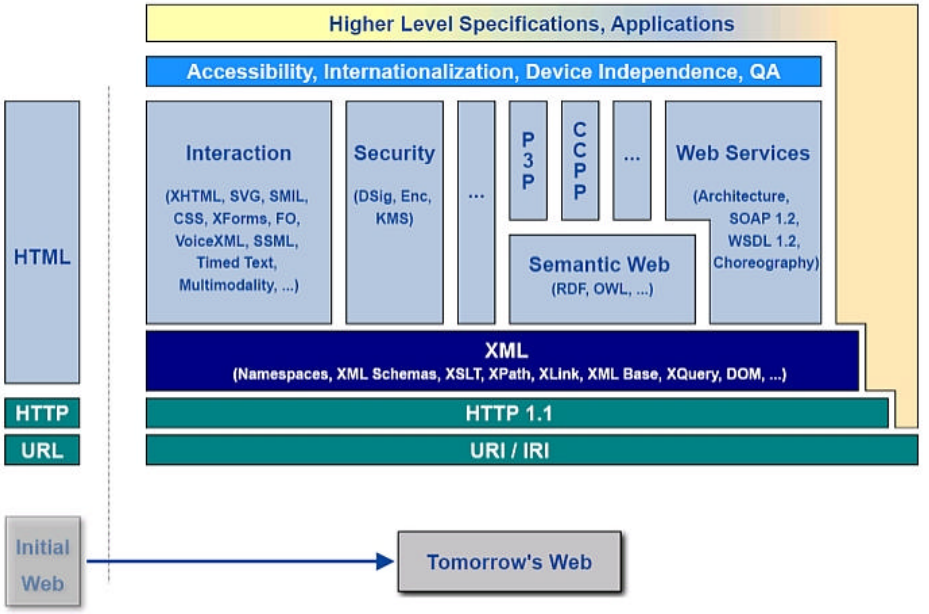

Ilustración 1. Esquema resumen de las especificaciones elaboradas por el W3C (extraído del sitio Web www.w3c.org).

2.1.7 Organización W3C

Para cumplir los objetivos propuestos (acceso universal, semántica Web y Web de confianza) mientras ejerce su papel (visión, diseño y estandarización) y aplica sus principios de diseño (interoperabilidad, evolución y descentralización), el proceso W3C está organizado de acuerdo a tres principios:

– Neutralidad. Los invitados del W3C (MIT, KEIO, ERCIM) son organizaciones neutrales, como lo es el Equipo. El W3C fomenta la

neutralidad facilitando y animando a la aportación durante todo el ciclo de vida de las especificaciones.

- Coordinación. La Web se ha convertido en un fenómeno tan importante (en alcance e inversión) que ninguna organización puede o debería tener control sobre su futuro. El W3C coordina sus esfuerzos con otras organizaciones de estandarización y consorcios como el IETF (Internet Engineering Task Force), el Unicode Consortium, el Web3D Consortium y varios comités ISO.
- Consenso. Este es uno de los principios más importantes por el cual el W3C opera. Cuando se resuelven temas o se realizan decisiones, el W3C trabaja para alcanzar una unanimidad de opinión. Cuando ésta no es posible, el W3C alcanza decisiones considerando las ideas y los puntos de vista de todos los participantes, sean Miembros del W3C, expertos invitados o el público general.

2.1.8 Equipo W3C

El Equipo W3C incluye investigadores e ingenieros de todo el mundo que lideran las Actividades técnicas en el W3C y gestionan las operaciones del Consorcio. La mayor parte de los Equipos de Trabajo se encuentran distribuidos físicamente entre alguna de las tres instituciones invitadas (MIT/LCS en los Estados Unidos, ERCIM en Francia y la Universidad Keio en Japón).

Liderado por el Jefe de Operaciones, Steve Bratt, y el Director, Tim Berners-Lee, el Equipo posee un número de roles que incluyen:

- Proporcionar una dirección al W3C para mantenerlo al tanto de las nuevas tecnologías, fluctuaciones del mercado y las actividades de las organizaciones relacionadas.
- Organizar y gestionar las Actividades del W3C como la optimización el proceso de alcanzar los objetivos dentro las habituales restricciones prácticas.
- Asegurar la cooperación entre los Miembros mientras fomenta la innovación.
- Mantener el sitio Web del Consorcio http://www.w3.org.
- Comunicar los resultados obtenidos por el W3C a los Miembros y la prensa.
- Comercializar los resultados del W3C para ganar su aceptación en la comunidad Web.
- Comercializar el W3C y atraer nuevos miembros ya que cuanto mayor sea la base de miembros más fácil será fomentar las Recomendaciones del W3C.

2.2 OASIS (Organization for the Advancement of Structured Information Standards)

OASIS en un consorcio global sin ánimo de lucro que conduce el desarrollo, convergencia y uso de estándares en el comercio electrónico (e-business). Los miembros mismos establecen la agenda técnica de OASIS usando un proceso abierto y ligero expresamente diseñado para producir estándares "mundiales" en seguridad, *servicios Web*, cumplimiento XML, transacciones de negocio, publicaciones

electrónicas, mapas temáticos e interoperabilidad dentro y entre mercados.

OASIS se componen de más de 600 corporaciones y miembros individuales ubicados en más 100 países repartidos por todo el mundo. OASIS y las Naciones Unidas conjuntamente patrocinan el desarrollo de ebXML, actúan como agencia central para esquemas de aplicaciones XML, vocabularios y documentos relacionados. OASIS hospeda las 'Cover Pages' o Páginas de Cobertura, una colección de referencias on-line para estándares de lenguajes de marca interoperables. La Red OASIS incluye UDDI, CGM Open, LegalXML y PKI.

OASIS se fundó en 1993 bajo el nombre de SGML Open como un consorcio de vendedores y usuarios dedicados a desarrollar guías básicas para la interoperabilidad entre productos que soportaran el 'Standard Generalized Markup Language' (SGML). OASIS modificó su nombre en 1998 para reflejar la extensión del alcance del trabajo técnico, incluyendo XML y otros estándares relacionados.

2.3 IBM/Microsoft/Verisign/RSA Security

Existe una colaboración entre varias compañías de gran relevancia en el ámbito de las IT encabezada por Microsoft e IBM.

En su documento (IBM & Microsoft, 2002b) proponen una pila de especificaciones relacionadas con la seguridad en servicios Web. En la base de esta pila de especificaciones se encuentra la especificación WS-Security estandarizada por OASIS.

Dicha pila de especificaciones está reflejada en la figura 2:

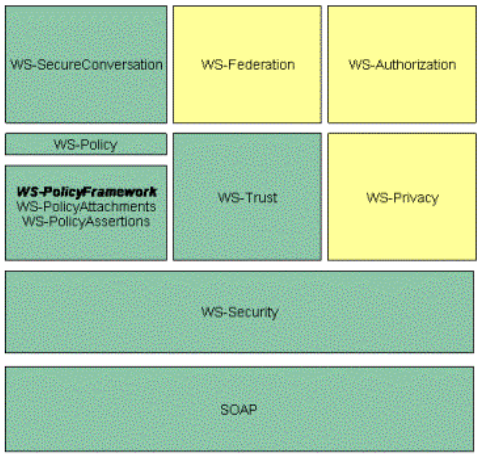

Ilustración 3. Familia de especificaciones de seguridad de la familia WS-*.

2.3.1 WS-Security

En la base nos encontramos con la especificación WS-Security. Esta especificación es un estándar según OASIS y proporciona integridad, confidencialidad y opcionalmente no repudio a los mensajes SOAP intercambiados entre servicios Web.

2.3.2 WS-Policy

Sobre esta especificación se encuentra el marco de trabajo que contiene las políticas para servicios Web. Con el fin de querer integrar con éxito con un servicio Web de cierta complejidad, debemos tener un amplio conocimiento del contrato XML del servicio además de cualquier requisito, capacidad, y preferencias adicionales (también llamado política). Por ejemplo, sólo conociendo que un servicio soporta WS-Security no es suficiente para conseguir la integración. El cliente necesita conocer si el servicio utiliza realmente WS-Security. En tal caso, también necesita conocer qué tokens de seguridad es capaz de procesar (tokens de seguridad Nombre de Usuario, tickets Kerberos o certificados), y cuál de ellos prefiere. El cliente debe determinar también si el servicio requiere o no mensajes firmados. Y, finalmente, el cliente debe determinar cuando cifrar los mensajes, qué algoritmos utilizar, y cómo intercambiar una clave secreta con el servicio. Intentar la integración con un servicio sin conocer todos estos detalles supone un altísimo gasto de esfuerzo del que será muy difícil obtener alguna recompensa. Sin una forma estándar de acordar las políticas, los desarrolladores quedan abandonados a su suerte y se ven en la situación de tener que desarrollar soluciones *ad-hoc* que no integren ni escalen.

Un marco de trabajo de políticas permitiría a los desarrolladores expresar las políticas de los servicios de una forma procesable por las computadoras. La infraestructura de los servicios Web puede verse ser mejorada para entender ciertas políticas y forzar su uso en tiempo de

ejecución. Por ejemplo, un desarrollador podría escribir una política que indique que un servicio dado requiere tickets Kerberos, firmas digitales y cifrado. Otros desarrolladores podrían utilizar la información de las políticas para determinar si pueden o no utilizar el servició, y todo ello en tiempo de ejecución sin la necesidad de ninguna intervención externa adicional. La infraestructura podría imponer estos requisitos sin obligar al desarrollador a tener que escribir ni una sola línea de código. Por lo tanto, un marco de trabajo de políticas no sólo provee una capa de descripción adicional, sino que también ofrecerá a los desarrolladores un modelo de programación más declarativo.

Microsoft, IBM, BEA, y SAP trabajan sobre esta especificación con el fin de obtener un marco de trabajo genérico de políticas. WS-Policy define un modelo genérico y una sintaxis para describir y comunicar las políticas de los servicios Web. Esta sección proporciona una perspectiva general de esta especificación y de otras estrechamente relacionadas.

2.3.3 WS-Trust

La especificación WS-Trust (S. Anderson, J. Bohren, T. Boubez, M. Chanliau, G. Della-Libera, B. Dixon, P. Garg, E. Gravengaard, M. Gudgin, P. Hallam-Baker et al., 2004) define extensiones que se construyen sobre WS-Security y que consisten en ampliar la capacidad de los mecanismos de seguridad definidos por ésta, permitiendo la **solicitud**, **emisión** e **intercambio** de tokens de seguridad y la gestión de las relaciones de confianza. Actualmente esta especificación se encuentra

en estado de borrador público inicial y su único propósito es ser objeto de revisión. El concepto de seguridad clave de esta especificación es la **confianza**.

Como ya hemos estudiado, WS-Security define los mecanismos básicos para proporcionar un marco de trabajo seguro en el intercambio de mensajes. Esta especificación, a partir de estos mecanismos básicos, define primitivas adicionales junto con extensiones para el intercambio de tokens de seguridad que permitan la emisión y propagación de las credenciales dentro de diferentes dominios de confianza.

Con el fin de garantizar la comunicación segura entre las partes, ambas deben intercambiar credenciales de seguridad (directa o indirectamente). Sin embargo, cada interlocutor necesita determinar si puede confiar en las credenciales presentadas por el otro.

Las extensiones que proporciona sobre WS-Security son:

- Métodos para *emitir*, *renovar* y *cambiar* (un tipo por otro) tokens de seguridad,
- Métodos para establecer y acceder a las relaciones de confianza presentes.

Mediante el uso de estas extensiones, la aplicaciones pueden dedicarse al diseño de las infraestructuras necesarias para obtener una comunicación segura que trabaje dentro del framework general de los servicios Web. Por ejemplo, incluyendo descripciones de servicios WSDL, estructuras

de entradas UDDI *businessServices* y *bindingTemplates*, y mensajes SOAP.

Para alcanzar este objetivo, esta especificación introduce un número de cabeceras SOAP y elementos que son utilizados para solicitar tokens de seguridad y gestionar las relaciones de confianza. El objetivo principal de esta especificación es, por lo tanto, habilitar a los sistemas para que puedan crear patrones de intercambio confiados de mensajes. Esta confianza se representa mediante el intercambio e intermediación de los tokens de seguridad. La especificación define un protocolo agnóstico que permite **emitir**, **renovar** e **intercambiar** tokens de seguridad.

Debemos dejar claro que ciertos temas como el establecimiento de un token de contexto de seguridad o la derivación e intercambio de claves quedan fuera del alcance de la especificación WS-Trust. De hecho, ambas características son desarrolladas en otra especificación de la familia WS-* denominada WS-SecureConversation y de la que también se ocupa este documento más adelante

2.3.4 WS-SecureConversation

WS-Security proporciona los mecanismos básicos sobre los que definir un marco de mensajería seguro. La especificación WS-SecureConversation (S. Anderson, J. Bohren, T. Boubez, M. Chanliau, G. Della-Libera, B. Dixon, P. Garg, E. Gravengaard, M. Gudgin, S. Hada et al., 2004) define un conjunto de extensiones para permitir el establecimiento, compartición y derivación de contextos y claves de

seguridad entre los interlocutores de una comunicación. Mientras que la especificación WS-Security se centra en un modelo de autenticación de mensajes, que aunque resultará muy útil en muchos casos se encuentra sujeto a múltiples formas de ataque, WS-SecureConversation introduce el concepto de seguridad contextual y describe como utilizarlo. Esto permite que se establezcan contextos y, potencialmente, que se realicen intercambios más eficientes de claves o de nuevo material de clave, consiguiendo el incremento global del rendimiento y de la seguridad en los intercambios posteriores.

El modelo de autenticación basado en el establecimiento de contextos de seguridad autentica una serie de mensajes requiriendo una serie de comunicaciones adicionales como paso previo a los intercambios de la información de negocio entre los servicios Web. Para implementar este modelo, WS-SecureConversation define una serie de cabeceras y extensiones nuevas del framework de mensajería SOAP.

Por lo tanto, y en función a lo dicho, las metas principales de WS-SecureConversation son:

- Definir cómo se establecen los contextos de seguridad.
- Especificar cómo se calculan y distribuyen las claves derivadas.

2.3.5 WS-Federation

Esta especificación (Bajaj et al., 2003) define los mecanismos necesarios para conseguir la federación de dominios de seguridad distintos o similares y lo consigue permitiendo e intermediando la confianza de las

identidades, atributos, y autenticación de los servicios Web participantes. La especificación WS-Federation define un modelo y un marco de trabajo general para conseguir la federación. Tal y como se expresa en la especificación, se irán generando documentos, denominados *profiles* (perfiles) los cuales detallarán cómo los distintos solicitantes encajan en este modelo.

Es por tanto un objetivo prioritario de esta especificación habilitar la federación de la información de las identidades, atributos, autenticación y autorización.

Los requisitos que conducen la elaboración de esta especificación son los siguientes:

- Habilitar una compartición apropiada de los datos de la identidad, atributos, autenticación y autorización utilizando diferentes mecanismos.
- Intermediación de la confianza e intercambios de tokens de seguridad.
- Las identidades locales de un dominio de seguridad no tienen por que encontrarse en los otros servicios con lo que se realizará la federación. Es decir, es requisito que mi dominio de seguridad *foo.com* defina un usuario A y que yo pueda federar su identidad y atributos con en otro dominio de seguridad *zyx.es* sin la necesidad de que éste último tenga que conocer la identidad del usuario A.

- Ocultamiento opcional de la información de la identidad y otros atributos. Es decir, el usuario A en el dominio *foo.com* puede acceder a distintos recursos en los dominios *abc.es* y *xyz.org* y su navegación no debería poder ser trazable a través de estos dominios y recursos ya que su identidad real, A, está oculta mediante algún mecanismo de alias o pseudónimos.

2.3.6 WS-Authorization

Esta especificación no dispone todavía de ninguna versión publicada. El principal objetivo de esta especificación será cómo definir y gestionar las políticas de control de acceso a los servicios Web.

2.3.7 WS-Privacy

Esta especificación, de la que todavía no existe una publicación, utiliza los mecanismos definidos por la especificación WS-Security, WS-Policy y WS-Trust para permitir la transmisión de las políticas de privacidad. Estas políticas de privacidad serán definidas por la organización propietaria del servicio Web y los mensajes SOAP recibidos deberán ser conformes con dichas políticas de privacidad. WS-Privacy señalará cómo incluir afirmaciones de política de privacidad mediante WS-Policy y utilizará WS-Trust para evaluar las sentencias de privacidad encapsuladas en los mensajes SOAP contra las preferencias de los usuarios y las políticas de la organización.

2.4 WS-I

La Organización para la Interoperabilidad de los Servicios Web es un esfuerzo de la industria con el objetivo de fomentar la interoperabilidad de los servicios Web entre distintas plataformas, aplicaciones y lenguajes de programación. La organización une una comunidad diversa de líderes en servicios Web y cuyo principal objetivo es dar respuesta a las necesidades de los clientes proporcionando unas directrices, prácticas recomendadas y recursos de ayuda para el desarrollo de servicios Web interoperables. Esta comunidad desarrolla activamente herramientas, recursos y otras guías de apoyo para ayudar en la implementación de los servicios Web.

2.4.1 La Comunidad WS-I

Los miembros de la comunidad WS-I incluye vendedores de software de todos los tamaños, empresas clientes, y otros muchos interesados en servicios Web. Todos los miembros de la comunidad están invitados a participar activamente en uno o más Grupos de Trabajo (Working Groups), en función a sus intereses y experiencia. Dependiendo del Grupo de Trabajo, el esfuerzo será dirigido por individuales que poseen un conjunto diverso de habilidades: desarrolladores, probadores, analistas de negocio y arquitectos estándar.

2.4.2 Entregables

Los entregables de la organización están destinados a proporcionar recursos para cualquier desarrollador de servicios Web que tenga la intención de crear servicios Web interoperables, y verificar que sus

resultados son acordes con los estándares de la industria y con las líneas básicas recomendadas por el WS-I. Los recursos más importantes que serán proporcionados serán herramientas que el WS-I denominan 'sniffers' y 'analizadores'. El proceso utilizado para desarrollar estas herramientas generará otros recursos de implementación útiles durante el camino.

- Perfiles. Conjunto de especificaciones sobre servicios Web que trabajan conjuntamente para soportar tipos específicos de soluciones.
- Implementaciones de ejemplo. Con el contexto de un perfil, los equipos trabajan para definir un conjunto de servicios Web que son implementados por múltiples miembros de distintos equipos y que identifican los puntos donde la interoperabilidad está presente.
- Directrices de implementación. Recomendaciones para el uso de especificaciones de maneras que hayan sido probadas ser las más interoperables.
- Rastreador. Son herramientas para monitorizar y registrar interacciones con un servicio Web. Estas herramientas generan un fichero que puede ser posteriormente procesado por un analizador.
- Analizador. Son herramientas que procesan registros generados por los rastreadores y que verifican que la implementación de un servicio Web está libre de errores.

2.4.3 Actividades

El trabajo del WS-I se realiza dentro de los Grupos de Trabajo. Cada Grupo de Trabajo es responsable de trabajar sobre un conjunto específico

de entregables. Algunos de los Equipos de Trabajo desarrollarán herramientas de prueba, otros proporcionarán servicios Web de ejemplo, etc. A medida que madure el trabajo de los Grupos de Trabajo se irá determinando la estructura exacta y la lista de estos.

2.4.4 Asociados

Existe una fuerte cooperación de este consorcio con otras organizaciones comprometidas al crecimiento de los servicios Web. Algunas organizaciones asociadas que complementan el actual trabajo del WS-I son con el W3C (World Wide Web Consortium) y el IETF (Internet Engineering Task Force).

WS-I proporciona una implementación de soporte como recurso de guía y pruebas para cada Perfil desarrollado. Organizaciones como el W3C sirve como fuente fundamental para las especificaciones que forman el núcleo de estos Perfiles WS-I, y sus miembros son los consumidores del material que el WS-I produce. Las especificaciones de estas organizaciones siguen siendo esenciales para el desarrollo de la World Wide Web.

El WS-I ha comenzado ha establecer enlaces con organizaciones relevantes y sus grupos de trabajo de especificación y arquitectura. El WS-I está comprometido en construir sólidas relaciones y a adoptar especificaciones desarrolladas por un amplio espectro de organizaciones tales como el IETF, OASIS, W3C y empresas clientes. Es un plan del

WS-I captar estos grupos y trabajar conjuntamente para satisfacer de la mejor manera posible las necesidades de sus clientes.

2.5 Liberty Alliance Project

El proyecto Liberty Alliance Project se formó en Septiembre del 2001 con el fin de desarrollar estándares abiertos de gestión de las identidades de red federadas y de servicios basados en identidad. Su principal objetivo es garantizar la interoperabilidad, proporcionar privacidad, y promover la adopción de sus especificaciones, líneas básicas de trabajo y buenas prácticas.

El grupo está formado por más de 150 miembros que representan organizaciones cuyo ámbito es muy diverso: institutos educacionales, organizaciones gubernamentales, proveedores de servicio, instituciones financieras, compañías IT, y proveedores de servicios.

El Liberty Alliance Project desarrolla especificaciones abiertas y no desarrolla productos o servicios específicos. La meta, como ya se ha dicho, es crear especificaciones que incorporen, aprovechen y soporten otros estándares de la industria, proporcionando un medio para que sus miembros y organizaciones externas puedan construir productos y servicios que interactúen, disminuyan el coste de propiedad y promuevan la gestión de la federación de las identidades de manera segura.

3. LA SEGURIDAD EN LA ARQUITECTURA DE REFERENCIA DE SERVICIOS WEB

3. La Seguridad en la Arquitectura de Referencia de los Servicios Web

En esta sección vamos a estudiar los requisitos de seguridad enumerados por la especificación que define la arquitectura de referencia de los servicios Web (W3C, 2004) y mencionaremos los distintos tipos de ataques, que también señala esta especificación, contra los que se verán expuestas las soluciones diseñadas basadas en servicios Web.

3.1 Servicios de seguridad básicos

La seguridad es un concepto más de entre aquellos servicios que comprenden la Calidad del Servicio (QoS) en los servicios Web (los otros tres son la transaccionalidad, fiabilidad, y gestión/administración) (Endrei et al., 2004a). Los servicios de seguridad básicos mencionados por la ISO 7498-2 son la confidencialidad, integridad, autenticidad de origen, no repudio y control de acceso y serán estos aspectos sobre los que centraremos el caso de estudio desarrollado en el siguiente apartado. Por su parte, la arquitectura de referencia planteada por el W3C para los servicios Web (W3C, 2004) hace mención a que para garantizar la seguridad en los servicios Web es necesario "un amplio espectro de mecanismos que solventen problemas como la autenticación, el control de acceso basado en roles, la aplicación efectiva de políticas de seguridad distribuidas o la seguridad a nivel de los mensajes". Definamos los servicios de seguridad básicos encontrados en cualquier sistema distribuido:

- Autenticación de los interlocutores. Cada servicio Web participante en una interacción podría requerir autenticación de la otra parte. Cuando cierto servicio A dirige una petición al servicio B, éste puede requerirle unas credenciales junto con una demostración de que le pertenecen como por ejemplo un par nombre de usuario (credencial)/password (demostración) o un certificado X.509v3 (credencial)/firma digital (demostración). El principal problema de la autenticación, cuyo origen radica en la naturaleza heterogénea de los servicios Web, es conseguir un acuerdo en los protocolos y en los formatos de los datos de seguridad empleados. Otro punto a resolver es definir un modelo de autenticación Single Sign-On de forma que un servicio Web A que necesita interactuar con otros 6 servicios Web, para completar un proceso de negocio P en tiempo real, no necesite autenticarse más que una vez frente al primero de ellos para poder completar la operación bajo un tiempo de respuesta aceptable.
- Autorización. Los servicios Web deben disponer de mecanismos que les permitan controlar el acceso a sus servicios (recursos). Se debe poder determinar quién y cómo puede hacer a qué y cómo sobre sus recursos. La autorización concede permisos de ejecución de ciertos tipos de operaciones sobre ciertos recursos a ciertas identidades autenticadas. Normalmente ese conjunto de restricciones que gobiernan el acceso a los recursos se materializan en forma de políticas de seguridad de acceso.

- Confidencialidad. Mantener de manera secreta la información crítica contenida en los mensajes intercambiados entre los servicios Web supone otra de las propiedades fundamentales que deben darse para que el canal de comunicación establecido se pueda considerar "seguro". La confidencialidad se consigue mediante técnicas de cifrado ampliamente ya utilizadas y extendidas en otros campos de la computación. La confidencialidad no sólo afecta al canal que utiliza para intercambiar los mensajes.
- Integridad. Esta propiedad garantiza a un servicio Web que la información que recibe es la misma que la información que fue enviada desde un sistema cliente.
- No repudio. Cuando se realizan transacciones suele ser un requisito ser capaz de probar que una acción tuvo lugar y que fue realizada por cierto actor. En el caso de los servicios Web, es necesario ser capaz de demostrar que un cliente utilizó un servicio pese a que éste lo niegue (no repudio del solicitante) así como demostrar que un servicio fue ejecutado (no repudio del receptor).
- Disponibilidad. La necesidad de cuidar el aspecto de disponibilidad, como prevenir ataques de denegación del servicio (DoS) o disponer de redundancia de los sistemas, es un punto crucial en la tecnología de los servicios Web sobre todo en aquellos casos en los que los servicios en cuestión son de alta criticidad: servicios en tiempo real, servicio de CRLs, etc.
- Auditabilidad. Los sistemas basados en servicios Web deben mantener una traza de todas las acciones que llevan a cabo de

forma que sea posible realizar un análisis posterior que permita averiguar, por ejemplo, lo ocurrido en escenarios de desastre.

- Seguridad extremo-a-extremo (Saltzer, Reed, & Clark, 1984). Las topologías de redes de servicios Web requieren la garantía de que la seguridad se mantenga a lo largo del recorrido seguido por los mensajes entre los dos extremos de la comunicación. El hecho de que puedan existir intermediarios en el camino del mensaje que puedan procesar parte del mismo exige un extra de seguridad que, no sólo garantice que el transporte entre los extremos y a través de los intermediarios es seguro, sino que además garantice la seguridad en cada nodo encontrado en el camino. Es necesario disponer de facilidades que permitan establecer un contexto de seguridad único y de extremo a extremo. El problema de convivir con varios contextos de seguridad en un canal de comunicación establecido entre dos interlocutores es que la información que es recibida y reenviada por un intermediario y que se modifica por encima de la capa de transporte puede perder la integridad o puede ser espiada. Para conseguir un único contexto de seguridad sobre la información intercambiada se deberá aplicar las correspondientes operaciones criptográficas sobre la información de aplicación en el contexto de la aplicación origen de forma que no se dependa de la seguridad configurada por debajo de la capa de aplicación (p.e.: a nivel IP mediante VPN o IPSec) para garantizar los servicios de seguridad básicos.

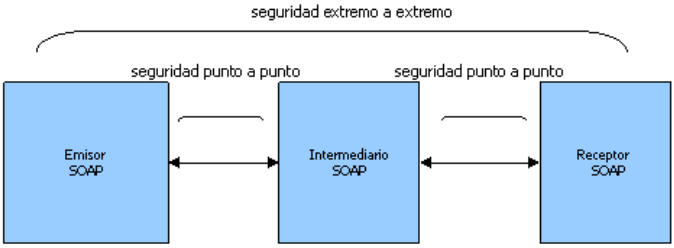

Ilustración 4. Seguridad punto a punto frente a la seguridad extremo a extremo.

3.2 Requisitos de seguridad

Cuando una organización debe decidir si adoptar o no una arquitectura basada en servicios Web debe tener en cuenta los numerosos desafíos que esto implica. Desde el punto de vista más abstracto, el objetivo es conseguir definir un entorno en el que las transacciones de los mensajes y los procesos de negocio se puedan llevar a cabo de manera segura de extremo a extremo. Una necesidad clara es la de asegurar los mensajes durante su tránsito, con o sin presencia de intermediarios. También existe la necesidad de garantizar la seguridad de los datos cuando son almacenados (control de acceso y privacidad). A continuación, se resumen los requisitos para poder proporcionar una garantía de la seguridad de extremo a extremo en los servicios Web:

- Mecanismos de autenticación. La autenticación es necesaria ya que permite verificar la identidad de los agentes solicitantes y proveedores (Lampson, Abadi, Burrows, & Wobber, 1992) (Polivy & Tamassia, 2002). En algunos casos, será necesario hacer uso de una autenticación mutua ya que los participantes no tendrán por qué estar conectados necesariamente de manera directa (podrían existir intermediarios en el camino que realicen ciertos procesos sobre los mensajes).

  En función de la política de seguridad en juego deberá ser posible autenticar al solicitante, al receptor o a ambos. Para poder llevar a cabo la autenticación se pueden emplear diversos métodos como contraseñas o certificados.

  Los métodos de autenticación basados en palabras de paso deberán hacer uso de contraseñas robustas. Hay que tener en cuenta que la autenticación basada en contraseña por sí sola podría resultar ser insuficiente pudiendo ser necesario, si la evaluación de las vulnerabilidades en el sistema así lo detecta, combinar métodos de autenticación por contraseña con otros tipos de autenticación o procesos de autorización como certificados, Directory Access Protocol (LDAP), Remote Authentication Dial-in User Service (RADIUS), Kerberos, o Infraestructuras de Clave Pública.

- Autorización. La autorización es necesaria si queremos controlar el acceso a los recursos. Una vez se ha autenticado al solicitante y se conoce su identidad, se utilizarán los mecanismos de autorización para controlar el acceso apropiado a los recursos del sistema (Anderson, Proctor, & Godik, 2004). Debería existir un acceso controlado a los sistemas y a sus componentes. Las políticas determinan los derechos de acceso de los solicitantes.

  Una de las ramas de investigación más interesantes de la seguridad en los servicios Web es la gestión y negociación de la confianza (Grandison, Sloman, & College, 2000). Este concepto permite que un cliente y un proveedor, que no poseen relación alguna previa, puedan, en función a credenciales que se intercambian, determinar el grado de confianza mutuo. La gestión y negociación de la confianza permite que las partes se autentiquen sin necesidad de que desvelen sus identidades.

- Integridad y confidencialidad de los datos. Las técnicas de integridad de los datos garantizan que la información no ha sido alterada, o modificada, durante la transmisión sin que este hecho sea detectado. La confidencialidad de los datos asegura que los datos sólo son accesibles por aquellas entidades determinadas para ello. Las técnicas más comunes aplicadas para este propósito son el cifrado de la información, cuando se trata de la confidencialidad, y la firma digital, cuando se trata de la integridad.

- No repudio. El no repudio es un servicio de seguridad que protege a un interlocutor que participa en una transacción contra la posible denegación, posiblemente por parte de alguno de sus participantes, de que la transacción ocurrió o de su participación en la misma.

  Las tecnologías de no repudio proporcionan evidencias sobre la ocurrencia de las transacciones de tal forma que una tercera parte, a partir de ellas, puede resolver el desacuerdo.

- Rastreabilidad. Las trazas son necesarias con el objeto poder conocer a posterior el acceso y comportamiento de los usuarios. También son necesarias si queremos garantizar la integridad del sistema a través de la verificación. Las trazas de auditoria pueden ser generadas por agentes. Estos agentes pueden tomar el papel de guardas de auditoria (concepto que explicaremos más tarde cuando veamos las dos categorías en las que engloba la arquitectura las políticas de los servicios Web) que pueden monitorizar; observar a los recursos y a otros agentes, validar que las obligaciones que han sido establecidas están siendo respetadas y descargadas. A menudo, no resulta posible prevenir la violación de las obligaciones. Sin embargo, si un guardia de auditoria detecta una violación de una política, podría tomar medidas cd manera activa.

- Aplicación distribuida de las políticas de seguridad. Aquellos que implementen una arquitectura basada en servicios Web deben ser capaces de definir una política de seguridad y de forzar su aplicación por múltiples plataformas con privilegios variables (Chang, Chen, & Hsu, 2003).

- Consideración de seguridad de la arquitectura. Las organizaciones que implementen los servicios Web deben ser capaces de ejecutar los procesos de negocio de una manera segura. Esto implica que todos los aspectos de los servicios Web, incluyendo enrutamiento, gestión, publicación y descubrimiento, debería ser llevado a cabo de una manera segura. Aquellos que implementen los servicios Web deben ser capaces de utilizar los servicios de seguridad como la autenticación, autorización, cifrado y auditoria.

Los mensajes de los servicios Web pueden atravesar los cortafuegos, y pueden ser "tuneleados" a través de puertos y protocolos existentes. La seguridad en los servicios Web requiere el uso de políticas corporativas que puedan ser integradas con políticas de otras empresas externas y con la resolución de la confianza. Las organizaciones podrían necesitar implementar las siguientes capacidades:

- Identidades en diversos dominios (Della-Libera et al., 2003) (LibertyAllianceProject, 2003). Los agentes solicitantes y proveedores se pueden comunicar entre sí utilizando diferentes

esquemas de identificación procedentes de diferentes dominios. Muchos sistemas definen los privilegios de acceso basado en roles en función de las identidades. Es importante que los servicios Web sean capaces de soportar la correspondencia de identidades a través de múltiples dominios, incluido un único dominio.

Una entidad proveedora y una entidad solicitante pueden usar sus identidades para cifrar y firmar los mensajes que intercambian. Además, pueden intercambiar credenciales de identidad dentro de un contexto de mensajes iniciales (fase de handshake). La identidad del servicio es opcional, y es perfectamente posible implementar un servicio de negocio sin una identidad si siempre actúa en beneficio de la entidad solicitante (es decir, adoptando la identidad del solicitante). El fenómeno contrario es más raro de ver ya que supondría no disponer de la identidad del solicitante, lo que se corresponde con un usuario anónimo, cosa que, en el contexto de los servicios de negocio, resulta complicado de encontrar.

- Políticas distribuidas. Las políticas de seguridad que se encuentran asociadas con una entidad solicitante o proveedora o con un mecanismo de descubrimiento, se puede utilizar para definir los privilegios de acceso de las peticiones y de las respuestas entre las partes. Estas políticas se pueden validar en

tiempo de ejecución en el contexto de una interacción. Cada parte en la interacción debe validar sus propias políticas.

- Políticas de confianza. Las políticas de confianza son políticas distribuidas que aplican al entorno de la otra parte de la comunicación en una interacción. Una entidad solicitante necesita confiar en el entorno del servicio. Las políticas de confianza pueden ser recursivas, es decir, pueden estar definidas contra las políticas de confianza de las partes implicadas e incluso contra los dominios de confianza completos. Un ejemplo sería la siguiente afirmación: “yo confío en ti si tú confías en mi amigo y mi amigo confía en ti”.

Las identidades, políticas y confianza distribuidas pueden estar descritas y pueden ser procesadas por máquinas. Por ejemplo, un certificado X.509 puede estar incrustado en un mensaje, de forma que afirma la identidad del emisor. Una política puede describir en XML y puede estar ligada al contrato con el servicio. Las máquinas podrían procesar, resolver y ajustar la seguridad en base a las descripciones otorgadas.

Los mecanismos de confianza se pueden utilizar para formas relaciones de delegación y federación. Estos mecanismos se pueden utilizar para facilitar interacciones seguras entre servicios Web que van más allá de los límites del dominio de confianza.

- Mecanismo de Descubrimiento Seguro. El mecanismo de descubrimiento seguro aplica políticas que gobiernan la publicación y descubrimiento de un servicio. Cuando se publica un servicio, normalmente se necesita una identidad para evaluar las políticas de publicación del servicio, excepto en aquellos casos de los servicios de descubrimientos entre nodos. Cuando un solicitante descubre un servicio, podría proporcionar o no una identidad, siendo en este segundo caso, un descubrimiento anónimo.

- Confianza y Descubrimiento. Supongamos que una entidad solicitante descubre un servicio Web que le interesa y que éste es ofrecido por una entidad proveedora hasta entonces desconocida para él. ¿Debería confiar la entidad solicitante en ese servicio? Esta pregunta se hace más profunda si el uso de ese servicio requiere que el solicitante divulgue información sensible ya que implica correr un riesgo muy alto.

  La dificultad en tomar esta decisión provoca una ramificación en el proceso de descubrimiento en función de si éste es manual o automático.

  Cuando el proceso de descubrimiento es manual, un humano podrá juzgar si se puede o no confiar en ese servicio. Sin embargo, en un proceso de descubrimiento automático, la máquina es la que debe tomar la decisión. Como los seres

humanos todavía no tenemos una confianza plena en el tipo de juicio que las máquinas pueden llevar a cabo, los agentes que realizan el descubrimiento autónomo se ven normalmente restringidos a utilizar servicios de descubrimiento privados que listan aquellos servicios que han sido preanalizados y que se han considerado de confianza.

Esta forma limitada de descubrimiento autónomo, se debería denominar de manera más precisa como selección autónoma, ya que los candidatos disponibles son conocidos por adelantado.

Dos formas de mitigar este aspecto de la confianza en los descubrimientos automatizados son:

- Que un agente pueda descubrir autónomamente los servicios Web candidatos y a continuación se los muestre un usuario humano para que escoja.
- Que un agente pueda descubrir autónomamente los servicios Web y verificar en un registro de confianza información independiente sobre ellos.

- Privacidad. La privacidad vista en términos de comportamientos, hábitos y acciones se expresan a partir de las políticas que los propietarios de los datos, normalmente los usuarios de los servicios Web, poseen, junto con los mecanismos necesarios que permiten garantizar que los derechos de los propietarios son respetados.

Las políticas de privacidad tienen por lo normal un componente de obligatoriedad mucho más alto que las políticas de control de acceso. Una política que requiere que un agente proveedor propague etiquetas P3P (Cranor, Langheinrich, Marchiori, Presler-Marshall, & Reagle, 2002), por ejemplo, representa una obligación sobre el proveedor de la entidad. Por tanto, y como no es posible saber si cierto agente proveedor está propagando la información de privacidad de la manera correcta, debe ser posible monitorizar las acciones públicas llevadas a cabo por el servicio Web con el fin de verificar que las etiquetas P3P están siendo distribuidas de la manera adecuada.

Muchas restricciones relacionadas con la privacidad guardan una relación estrecha con el mantenimiento de ciertos tipos de estado. Por ejemplo, una entidad proveedora podría tener una restricción de forma que cualquier etiqueta P3P asociada con un uso de alguno de sus servicios Web debe ser apropiadamente propagada a terceros. Tal restricción no se puede expresar fácilmente en términos de las acciones permitidas que el agente proveedor podría realizar. Es una obligación garantizar que la condición observable pública (el uso adecuado de las etiquetas P3P) se mantiene siempre.

De la misma manera un agente proveedor podría asociar las acciones posibles que cierto agente solicitante podría realizar al

mantenimiento que éste haga de un nivel particular de acceso seguro (p.e.: las tareas administrativas sólo se pueden llevar a cabo si la petición está utilizando una comunicación segura).

- Fiabilidad de los servicios Web (Zhang, Zhang, & Chung, 2004). El hecho de tener que tratar con posibles errores es un hecho del que no es posible escapar, específicamente en el contexto de una red global de servicios conectados que pertenecen a muchas personas, entidades, u organizaciones diferentes. Como los errores no se pueden eliminar un 100%, la meta debe ser reducir la frecuencia de error al máximo durante las interacciones y, cuando ocurra un error, proporcionar una gran cantidad de información.

  En los servicios Web, el aspecto de la fiabilidad se puede distribuir por distintos niveles según el nivel de fiabilidad y predecibilidad de:

  - Los servicios de la infraestructura, como un mecanismo de transporte de los mensajes o un servicio de descubrimiento.
  - Las interacciones entre los servicios.
  - El comportamiento individual del solicitante y del proveedor.

En cualquier sistema distribuido existen límites fundamentales a la fiabilidad de las comunicaciones entre los agentes de una red pública. Sin embargo, en la práctica existen técnicas que se pueden utilizar para incrementar considerablemente la fiabilidad de la mensajería y, en aquellos casos en los que la comunicación falle, se puede obtener la información que permita conocer la naturaleza del error.

Más detalladamente, podemos identificar dos propiedades de la fiabilidad en el envío de los mensajes que resultan muy importantes:

- El emisor del mensaje querrá saber si su mensaje fue recibido correctamente por el receptor
- El emisor del mensaje querrá saber si el mensaje fue recibido una sola vez.

Saber si un mensaje ha sido recibido correctamente o no permite al emisor tomar medidas compensatorias en el segundo de los casos. A lo sumo, el emisor podría intentar reenviar un mensaje que sabe que no fue recibido.

La meta general de la mensajería fiable es definir mecanismos que hagan posible alcanzar estos objetivos con una alta probabilidad de éxito sobre escenarios en los que ocurren errores de red, sistema o software impredecibles.

Las tres principales cuestiones que podríamos preguntarnos sobre la validez de un mensaje son:

- Si el mensaje recibido es idéntico al mensaje enviado. Esto se puede determinar mediante técnicas de conteo de bytes, sumas de verificación y firmas digitales.
- Si el mensaje es conforme con los formatos especificados por el protocolo acordado para el mensaje. Normalmente, esto se puede determinar por sistemas automáticos que utilicen restricciones sintácticas (p.e.: XML bien formado) y restricciones estructurales (validar el mensaje contra uno o más esquemas XML o definiciones de mensajes WSDL).
- Si el mensaje es o no conforme con las reglas de negocio esperadas por el receptor. Con este propósito, la lógica de aplicación y/o los gestores de procesos humanos verifican normalmente restricciones y comprobaciones relacionadas con el procesos de negocio

De entre todas éstas, la primera se considera resuelta de manera parcial por la mensajería fiable y la última está parcialmente solucionada por la teoría de coreografía de servicios.

La arquitectura de los servicios Web no ofrece un soporte específico para la mensajería fiable o para informar en el caso de que ocurra un fallo. Sin embargo, sí ofrece un guía para abarcar

ambos aspectos. Esta guía se basa en el uso de las cabeceras y el cuerpo de los mensajes: proporcionando cabeceras estándar que permiten la auditoria de los mensajes de forma que las infraestructuras de fiabilidad de los mensajes puedan ser desplegadas en formas que no provoquen ningún impacto en las aplicaciones y los servicios. Es posible así mejorar el tráfico de los mensajes según sea necesario mediante cabeceras específicas e intermediarios que implementen semánticas específicas de fiabilidad de la mensajería y 'reporting' en el caso de que la comunicación de un mensaje falle.

La fiabilidad de los mensajes se consigue vía una infraestructura de acuses de recibo que defina un conjunto de reglas que indiquen cómo se deben comunicar las partes con respecto a la recepción de ese mensaje y de su validez.

Las especificaciones WS-Reliability (Evans et al., 2003) y WS-ReliableMessaging (IBM, Microsoft, & TIBCO, 2004) son un ejemplo del uso de una infraestructura basada en acuses de recibo que aprovechan el modelo de Extensibilidad de SOAP. En aquellos casos en los que la capa de transporte subyacente ya proporcione el soporte para mensajería fiable (p.e.: una infraestructura basada en colas o HTTP-R, (Todd, Parr, & Conner, 2002)), se podrá alcanzar el mismo nivel de fiabilidad con SOAP mediante la definición de un vínculo que confíe en las propiedades subyacentes del mecanismo de transporte.

Otro concepto relacionado con la fiabilidad es la fiabilidad de los servicios. Al igual que ocurre con la fiabilidad de los mensajes, resulta imposible ofrecer una garantía de que el agente proveedor o el agente solicitante del servicio siempre actúe de manera correcta. De nuevo, y especialmente en el contexto de un sistema distribuido desplegado sobre una red pública en la que los agentes diferentes podrían pertenecer a diferentes entidades y podrían estar sujetos a políticas y funciones de gestión distintas, no resulta posible diseñar una solución completa que garantice la fiabilidad de un servicio. Sin embargo, podemos desplegar técnicas que pueden mejorar enormemente la fiabilidad y reducir el coste de los posibles fallos. La técnica principal es la gestión de un contexto transaccional (Cabrera et al., 2004). La gestión de la transacción permite que las conversaciones entre los agentes sean gestionadas de forma que todas las partes implicadas tengan un grado superior de seguridad de que las transacciones entre ellos progresan adecuadamente y, en el caso de que ocurriera un error, que el fallo será identificado y las transacciones serán canceladas, devueltas a su punto inicial o de alguna forma compensadas.

La arquitectura no da una advertencia específica sobre cómo implementar la fiabilidad transaccional. Sin embargo, la combinación de estructuras de mensajes flexibles y extensibles y el concepto de procesamiento múltiple de mensajes (vía

intermediarios que implementan roles de servicios) sí nos puede proporcionar un punto de partida.

Una manera de incorporar la transaccionalidad es utilizando cabeceras estándar que contengan información tipo marcadores transaccionales o informa ción contextual. Esta información puede ser incorporada a los mensajes intercambiados entre los agentes solicitantes y los proveedores de tal forma que los intermediarios pueden procesar los mensajes y pueden monitorizar las transacciones con un mínimo impacto en las aplicaciones existentes. Los intermediarios transaccionales especializados podrían procesar estas cabeceras específicas de transacción (como por ejemplo el comienzo de una transacción, 'commitment' o 'rollback') y marcar los mensajes que procesen con los resultados de forma que las aplicaciones puedan responder apropiadamente.

Relacionado con la monitorización transaccional se encuentra el concepto de monitorización de coreografías de los servicios. Un aspecto significativo de la especificación de la interfaz de un servicio es el patrón de mensajes que uno puede observar. En los casos más sencillos, este patrón es a menudo muy directo, como por ejemplo una interacción petición/respuesta; sin embargo, en casos más realistas, la coreografía de los servicio puede ser muy compleja. Monitorizar que los mensajes están llegando en el orden esperado es potencialmente una herramienta significativa en el despliegue de los servicios fiables.

De nuevo, y como ocurría con la monitorización transaccional, una aproximación es desplegar procesos intermediarios especializados cuya función específica sea asegurar que la coreografía y los requisitos relacionados con la estructura de los mensajes de un servicio se están cumpliendo. Esto resulta ser muy relevante cuando el agente proveedor de un servicio no es el mismo propietario del dominio que el del agente solicitante.

Las amenazas de seguridad tienden a tener asociadas los conceptos de acceso y uso de los recursos. Se debe garantizar que no va a ser posible para un intruso acceder a aquellos recursos para los cuales no posee los derechos de acceso apropiados.

Las principales amenazas se pueden resumir como:

- Acceso inapropiado en favor de entidades no identificadas correctamente. Debería ser posible determinar de manera confiable la identidad de los agentes, proveedores de servicio, recursos, etc.
- Corrupción del canal de comunicación y de los recursos. Debería ser posible garantizar la integridad de los datos de las comunicaciones y las transacciones.
- Debería también garantizarse que la información es sólo accesible por las partes deseadas. Esta certeza debe incluir el contenido de los mensajes; pero podría también incluir el propio hecho de que tuvo lugar una comunicación entre dos partes. Es decir, se podría desear que el mero hecho de la comunicación se produzca de

forma privada sin que terceras partes tengan conocimiento de ello.

- El acceso inapropiado a los recursos. Debería ser posible garantizar que los recursos y las acciones no son accesibles por aquellas partes que no están autorizadas. De nuevo, este hecho se podría extender al mero conocimiento de que cierto recurso existe, es decir, de alguna forma se debería poder impedir que personas no autorizadas no puedan conocer la existencia de ciertos recursos o ciertos servicios.
- Denegación de servicio. No debería ser posible que los clientes de los servicios no puedan acceder a él.

Debemos destacar que existen ciertas limitaciones teóricas que nos impedirán garantizar el desarrollo de contramedidas que evitan por completo estos riesgos. Sin embargo, debe ser una meta prioritaria conseguir la mitigación de estas potenciales brechas de seguridad de la arquitectura.

Existen otras amenazas que el diseño de la arquitectura WSRA propuesta por el W3C no tiene intención de resolver:

- No repudio. El riesgo asociado con este tipo de amenaza es que un participante de una transacción niegue a posteriori haber participado. Existen dos formas de no repudio posible. Cuando es el cliente del servicio Web el que niega haber realizado cierta transacción se denomina no repudio del cliente. Una amenaza de

este tipo es crítica por ejemplo en transacciones bancarias. Debería ser posible demostrar que cierto cliente de un servicio Web ordenó cierta transacción pese a que éste lo niegue. De la misma forma, el repudio del servicio impediría al proveedor de un servicio negar la consecución de ciertas acciones que él dijo en su momento haber llevado a cabo. Imaginemos que el servicio bancario a través del cual realizamos cierta transacción bancaria nos confirmó su consecución pero no tenemos capacidad para demostrar que, no solamente nos lo dijo, sino que además fue él el que realmente nos lo dijo.

- Información errónea. Una parte atacante podría intentar corromper un servicio Web mediante el envío, de forma deliberada, de información incorrecta. Por ejemplo, un solicitante de un servicio Web podría ganar acceso a cierto servicio mediante el envío de información incorrecta de una tarjeta de crédito. Este tipo de ataque se parece al desbordamiento de buffer salvo que la información enviada está bien formada aunque es falsa.
- Copia y repetición. Dada una copia de un agente servicio Web, podría ser posible ejecutarlo en un entorno que pudiera proporcionar a un atacante información a la que normalmente no tendría acceso. Por ejemplo, dada una copia de un agente solicitante de un servicio Web que actúa en favor de un cliente, un proveedor de un servicio Web atacante podría simular una serie de transacciones con el agente capturado con el fin de determinar el precio máximo que el usuario final estaría dispuesto a pagar por cierto bienes o servicios.

Los puntos de privacidad están relacionados con el uso de la información personal, en particular, con el abuso de la misma; de nuevo, esto puede ser expresado muy a menudo en términos de gente incorrecta accediendo de manera incorrecta a la información. Las amenazas relacionadas con la privacidad se podrían resumir en dos:

- Uso de la información. Un usuario final podría tener el derecho de conocer cómo, cuándo y con qué límite será utilizada su información personal o crítica por los servicios Web. La privacidad incluye la acción de compartir la información personal o crítica obtenida por un servicio Web con terceras partes.
- Confidencialidad. Esta amenaza, tal y como se explicó anteriormente, implica el riesgo de que terceras partes tengan acceso a la información crítica intercambiada a través del canal de comunicación establecido entre dos servicios Web.

Un punto también muy importante es que estas prácticas deberían ser expuestas de alguna manera por los servicios Web de manera previa a la invocación del servicio, permitiendo así al cliente del mismo, ejecutar su propia lógica de privacidad en función a la política de seguridad publicada por el servicio Web. Por lo tanto, la publicación y accesibilidad de las prácticas de seguridad, en cuestión de privacidad, aplicadas por un servicio Web permitirán cierto grado de control sobre su información personal a un usuario final.

3.3 Tipos de ataques

Los tipos de ataques que se listan a continuación están extraídos de la especificación (W3C, 2004).

- Alteración de los mensajes. Este tipo de amenazas afectan a la integridad de los mensajes, en tanto en cuanto un atacante podría modificar partes (o la totalidad) de un mensaje. Un atacante que tuviera interceptada la línea de comunicaciones entre los servicios podría hacer rastreo (sniffing), capturar los mensajes, eliminar o modificar parte o la totalidad del mensaje, y volverlo a poner en la línea de comunicaciones. La integridad de los mensajes también se verá afectada si el atacante realiza modificaciones sobre sus anexos.
- Ataques a la confidencialidad. Este tipo de amenaza se lleva a cabo cuando el atacante es capaz de interceptar los mensajes y de obtener, total o parcialmente, la información que transportan. Imaginemos lo preocupante que podría ser que un atacante interceptará los datos de la tarjeta de crédito con la que estamos pagando la reserva de un vuelo.
- Hombre en el medio o 'man-in-the-middle'. Los ataques 'man-in-the-middle' consisten en que un atacante se sitúa entre dos interlocutores e intercepta y, es capaz de responder, a los mensajes que éstos se están intercambiando dando la sensación a los interlocutores de que realmente se están comunicando entre sí cuando realmente lo están haciendo con el atacante. La siguiente figura muestra un ejemplo de este tipo de ataque:

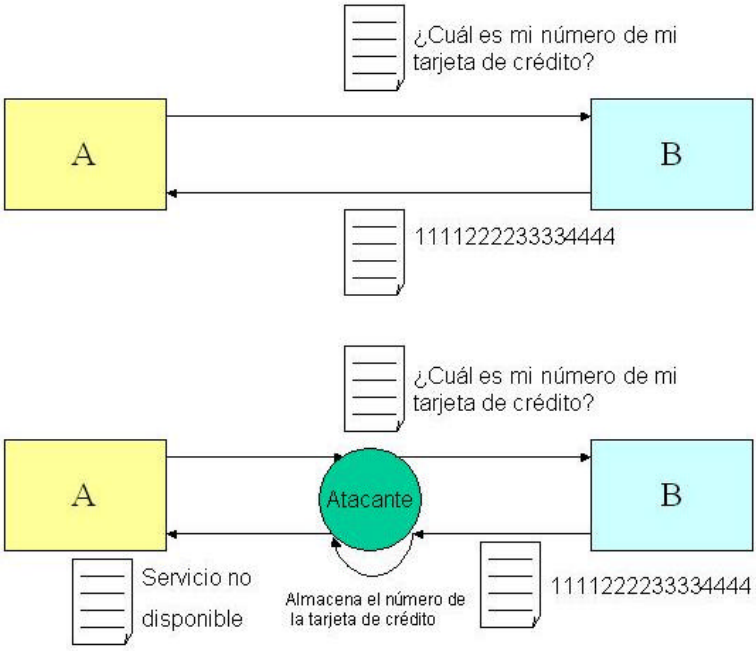

Ilustración 5. Ataque 'man-in-the-middle'

La entidad A realiza una petición solicitando al servicio B que le devuelve los datos de su tarjeta de crédito. El servicio B recibe la petición de A, verifica a partir de la información en el mensaje que es realmente A y le devuelve los datos de su tarjeta de crédito. En el escenario inferior vemos cómo se ha situado un atacante en medio de las comunicaciones. La petición realizada por A es interceptada por el atacante que simplemente la re-envía a B, la respuesta B también es interceptada por A. En este caso lo que hace el atacante es guardarse los datos de la tarjeta de crédito

y suplanta a B desde el punto de vista de A enviándole a éste último una respuesta indicativa, por ejemplo, de que el servicio se no se encuentra disponible en ese momento (cuando realmente sí lo está).

El atacante podría simplemente tener acceso a los mensajes (ataque pasivo) aunque también podría modificarlos (ataque activo). Para solucionar este tipo de ataques se suelen emplear técnicas de autenticación mutua.

- *Spoofing.* El *spoofing* es un ataque complejo que explota las relaciones de confianza establecidas entre los interlocutores. El atacante es capaz de suplantar la identidad de una de las partes de una relación de confianza con el objeto, normalmente, de comprometer la seguridad de la parte destinataria. En este ataque la parte destinataria siempre pensará que está dialogando con un interlocutor de confianza. Normalmente, esta técnica se utiliza como medio para realizar otros tipos de ataques como el 'forjado' de los mensajes. Se deben aplicar técnicas robustas de autenticación para poder hacer frente a este tipo de ataques.
- Denegación de servicio. Los ataques de denegación de servicio, comúnmente conocidos con el acrónimo DoS (Denial-of-Service) se centra en prevenir que los usuarios legítimos de un servicio sean capaces de hacer uso de él. Estos ataques suelen estar destinados a explotar el servicio de seguridad básico conocido como la disponibilidad. Los ataques de DoS pueden interrumpir las operaciones llevadas a cabo por un servicio consiguiendo su

desconexión efectiva respecto del resto del mundo. Los ataques por DoS pueden tomar varias formas, desde un intruso interno, con acceso físico a la máquina que proporciona el servicio, que desconecta un cable o para cierto proceso, a la realización de un sencillo 'script' que realice peticiones masivas hacia un mismo servicio de forma que es capaz de saturarlo impidiendo que ofrezca un servicio de calidad al resto de los usuarios (o incluso consiguiendo llegar a pararlo).

- Ataque de repetición. En este caso un atacante es capaz de interceptar un mensaje válido pudiendo re-enviarlo más tarde todas las veces que quiera al servicio para el que era destinatario. Para poder solventar este problema se deben utilizar técnicas de autenticación apropiadas conjuntamente con técnicas de sellado de tiempos y números de secuencia.

4. APLICACIÓN DE LOS ESTÁNDARES DE SEGURIDAD ACTUALES EN SERVICIOS WEB

4. Aplicación de los Estándares de Seguridad Actuales en servicios Web

Tal y como se menciona en el borrador de la especificación de la arquitectura de referencia de los servicios Web propuesto por el W3C "un servicio Web es una pieza software identificada por una URI, cuyas interfaces y vínculos se pueden definir, describir y descubrir como artefactos XML. Un servicio Web soporta interacciones directas con otros agentes software utilizando mensajes XML intercambiados mediante protocolos de Internet" (W3CWebServicesArchitectureWorkingGroup, 2004).

Los servicios Web son un paradigma de computación distribuida cuya amplia aceptación lo convierten en una implementación de gran relevancia de arquitectura orientada a servicios SOA (Service Oriented Architecture) (Erl, 2004). Las principales características de los servicios Web es que son piezas software auto-contenidas, auto-descritas, modulares, estructuradas a partir del ensamblaje de componentes, dinámicas, pueden ser publicados, localizados e invocados a través de la Web, son independientes del lenguaje de implementación e interoperables, independientes de la plataforma e inherentemente basados en estándares (Endrei et al., 2004b).

La seguridad es un concepto más de entre aquellos servicios que comprenden la Calidad del Servicio (QoS) de los servicios Web (los

otros tres son la transaccionalidad, fiabilidad, y gestión/administración). Los servicios de seguridad básicos mencionados por la ISO 7498-2 son la confidencialidad, integridad, autenticidad de origen, no repudio y control de acceso y serán estos aspectos sobre los que centraremos el caso de estudio desarrollado en el siguiente apartado. Por su parte, la arquitectura de referencia planteada por el W3C para los servicios Web (W3CWebServicesArchitectureWorkingGroup, 2004) hace mención a que para garantizar la seguridad en los servicios Web es necesario un amplio espectro de mecanismos que solventen problemas como la autenticación, el control de acceso basado en roles, la aplicación efectiva de políticas de seguridad distribuidas o la seguridad a nivel de los mensajes.

4.1.1 Caso de estudio

En nuestro caso de estudio tenemos a un usuario, Pedro, que posee una cuenta de usuario en el sitio Web, www.example.com. Este sitio Web es una Web tipo agencia de viajes que actúa como intermediaria entre el cliente y sitios Web especializados en reserva de vuelos y de coches. El usuario puede realizar dos operaciones en www.example.com: consultar itinerarios y reservar vuelos y coches para un itinerario dado. En el primer caso de uso, el usuario envía una petición con un itinerario y el sistema le devuelve distintos presupuestos en función de la petición realizada. En el segundo caso de uso el usuario envía, además del itinerario seleccionado, cierta información personal junto con información del medio de pago que utiliza (p.e.: los datos de su tarjeta de crédito). Como medida de seguridad básica y tradicional, el sitio Web

www.example.com ofrece un servicio HTTPS para el segundo caso de uso.

www.example.com dispone de acuerdos comerciales con sitios Web especializados en cada uno de los productos que vende a sus usuarios finales: www.example-coches.com para el alquiler de coches y www.example-vuelos.com para la reserva de vuelos. Además www.example.com utiliza n pasarelas de pagos cada una de ellas ubicadas en su servicio www.example-pagos-**n**.com.

El sitio Web www.example.com puede disponer de un repositorio UDDI (Atkinson et al., 2003) donde almacena la información de los servicios Web que hasta la fecha dispone para proporcionar sus servicios. Cada vez que llegue a un acuerdo con un nuevo sitio Web podría dar de alta este servicio en su UDDI privado para su posterior uso. Todos aquellos servicios Web nuevos que se vayan incorporando, que alquilen coches o reserven vuelos deberán aceptar el contrato semántico (W3CWebServicesArchitectureWorkingGroup, 2004) y deberán ofrecer sus servicios mediante la interfaz Web definida por www.example.com (que podría proporcionar 'off-line' mediante un fichero WSDL). Asumimos que www.example.com acepta las políticas de seguridad establecidas por los servicios Web que utiliza (y de las que puede disponer a través de su directorio UDDI). El propósito de este artículo no es conocer que puntos de seguridad y qué contramedidas podemos tomar en el proceso de publicación y descubrimiento de un servicio Web tema que, por sí mismo, posee entidad suficiente como para que pueda ser

tratado en otro artículo. En la siguiente figura se muestra el esquema básico sobre el que iremos desarrollando la discusión:

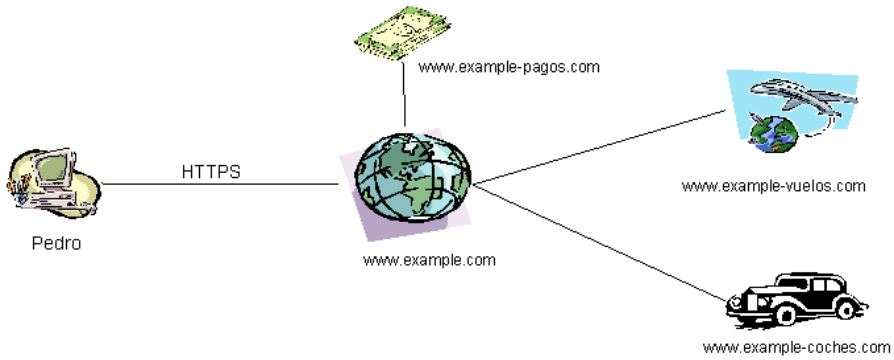

Ilustración 6. Esquema básico del caso de estudio.

Salvo entre Pedro y www.example.com que es puramente HTTPS, todas las conexiones reflejadas entre los distintos elementos suponen diálogos llevados a cabo a través de Internet mediante mensajes SOAP sobre HTTPS que contienen los datos de negocio en formato XML.

4.1.2 Seguridad a nivel de transporte

La primera medida de seguridad empleada por www.example.com para intercambiar mensajes SOAP a través de HTTP es utilizar SSL/TLS. Ambos protocolos nos permiten garantizar autenticación de las partes y la integridad y la confidencialidad de los mensajes en comunicaciones punto a punto. Sin embargo adoptar sólo SSL/TLS no es suficiente para garantizar la identidad de las partes, la confidencialidad y la integridad de la comunicación extremo a extremo a través de nodos intermediarios, estructura natural de las arquitecturas basadas en servicios Web. Como

IBM expresa, “cuando los datos son recibidos y reenviados por un intermediario por encima de la capa de transporte, tanto la integridad de los datos como cualquier otro tipo de información que contenga, podría perderse. Esto fuerza a que cualquier procesador de mensajes posterior confíe en las evaluaciones de seguridad hechas por los intermediarios previos y, a que tenga confianza plena en la forma en que éstos últimos hayan manipulado el contenido de los mensajes“(IBM & Microsoft, 2002a).

Además de emplear HTTPS como ‘binding’ SOAP (W3CXMLProtocolWorkingGroup, 2003), www.example.com y los servicios Web con los que interactúa podrían acordar el uso de mecanismos conformes con las especificaciones HTTPR (Todd et al., 2002), WS-ReliableMessaging (Bilorusets et al., 2004), o WS-Reliability (Evans et al., 2003). Estos mecanismos garantizan de una forma u otra la confiabilidad en la entrega de los mensajes intercambiados (el equivalente a los servicios de confiabilidad proporcionados por TCP al nivel de red IP en el modelo OSI). En la figura 2 vemos el resultado de aplicar algunos de estos mecanismos en los diálogos Web establecidos:

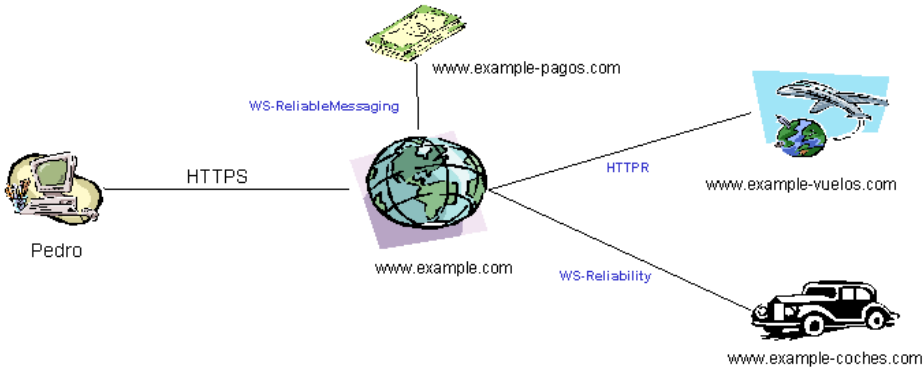

Ilustración 7. Aplicación de mecanismos de fiabilidad en el transporte.

Autenticación

En este caso vamos a estudiar la autenticación entre el servicio Web cliente de www.example.com y cualquiera de los servicios con los que interactúa. La manera más sencilla de autenticar a www.example.com es mantener una clave secreta (tipo usuario/contraseña) con cada unos de los servicios Web y enviársela de manera segura. Para solucionar este problema, www.example.com propone utilizar los mecanismos descritos en la especificación WS-Security (OASISWebServicesSecurityTC, 2004) en su documento SOAP Message Security 1.0 y emplear elementos de seguridad del tipo *UsernameToken* definidos en el perfil *Username Token Profile*. Esta especificación, cuyo proceso de estandarización está a cargo de OASIS (http://www.oasis-open.org) y cuya primera versión ha sido liberada en Abril del 2004, permite asegurar los mensajes SOAP

indicando una manera de aplicar las primitivas de seguridad XML de firma digital y cifrado según se expresa en los estándares liberados por el W3C (http://www.w3c.org) XML Digital Signature y XML Encryption. De esta forma www.example.com extiende el marco de trabajo de mensajería distribuido SOAP mediante un módulo SOAP que incluye una cabecera de seguridad con el nombre de usuario y clave secreta del sitio Web www.example.com. En cada mensaje enviado desde www.example.com hacia algunos de los servicios Web deberá enviar su clave/password de forma que demuestre su identidad. Análogamente, los servicios Web deberán utilizar algún mecanismo para demostrar la identidad del origen de las respuestas. Una alternativa podría ser aplicar firmas digitales. Si www.example.com firma digitalmente cierta parte del mensaje, los servicios Web receptores podrán validar la firma y comprobar así su identidad. Suponiendo que existe una tercera parte de confianza (o que las partes admitieran certificados auto-firmados) en la que todos los servicios involucrados confían, sería muy sencillo aplicar los mecanismos descritos por WS-Security y su perfil X.509 Certificate Token Profile en el esquema presentado. Hoy en día la aplicación de firmas digitales en el panorama XML está estandarizada por el W3C mediante la recomendación XML Digital Signature.

Ilustración 8. Aplicación de autenticación basada en certificados x509v3.

Integridad

La integridad de los mensajes garantiza que no han sido alterados durante el tránsito entre las partes. Cuando trabajamos con claves públicas/privadas y certificados, la forma clásica de solucionar esta problemática consiste en cifrar con la clave privada el resultado de aplicar una función resumen sobre el contenido del que queremos garantizar la integridad. Una vez más, podríamos aplicar las primitivas definidas en XML Digital Signature según lo indica el estándar WS-Security.

Confidencialidad

La confidencialidad de los mensajes se alcanza aplicando técnicas de cifrado sobre aquellas partes que deseamos mantener confidenciales frente a los posibles atacantes. La especificación XML Encryption (Imamura, Dillaway, & Simon, 2002), definida por el W3C, define un

modelo de procesamiento para cifrar, descifrar y formatear en XML datos cifrados. La manera de aplicar esta especificación sobre mensajes SOAP viene determinada por la especificación SOAP Message Security 1.0 que, como ya se ha mencionado, forma parte del estándar WS-Security 1.0.

La comunicación entre el portal www.example.com y www.example-pagos.com es un claro ejemplo en el que desearemos aplicar confidencialidad (e integridad) a ciertas partes de los mensajes. La figura 4 muestra los estándares que deberán emplear ambos portales si desean que exista una autenticación mutua entre las partes, así como integridad y confidencialidad en los mensajes que se intercambian. En este caso se están cifrando los mensajes mediante una clave simétrica conocida por ambas partes y distribuida inicialmente por procedimientos 'off-line'. La autenticación de las partes y la integridad de los mensajes se consiguen empleando firmas digitales.

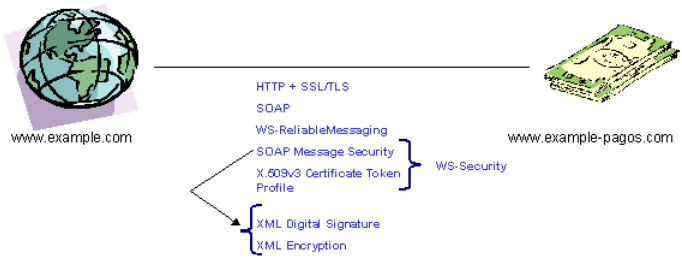

Ilustración 9. Autenticación, integridad y confidencialidad de los mensajes.

No repudio

La aplicación de firmas digitales y su correspondiente soporte legal permiten garantizar la irrenunciabilidad en los mensajes intercambiados entre los servicios Web. Por ello, se debe utilizar los mecanismos descritos en WS-Security para la aplicación de las firmas digitales mediante XML Digital Signature ofreciendo así una solución estándar a este servicio de seguridad.

Control de Acceso

El control de acceso hoy en día está estandarizado gracias al estándar XACML cuyo proceso de estandarización corre a cargo del consorcio OASIS. XACML permite definir políticas de control de acceso a los servicios Web. Existe un marco de trabajo alternativo compuesto de una serie de especificaciones, que todavía no han pasado por ningún cuerpo principal de estandarización, denominado WS-Policy Framework [16] cuyos autores son, entre otros, IBM y Microsoft.

Si nos ajustamos tan sólo a la especificación XACML podría interesarnos introducir cierta lógica de control de acceso si nuestro sistema es www.example-vuelos.com de forma que podamos restringir que las peticiones procedentes de www.example.com sólo sea procesadas si se reciben entre las 10:00 a.m. y las 20:00 p.m de lunes a sábado y, además, que estas peticiones sólo pueden ser de reserva de plazas de avión en clase turista. Con XACML es posible crear políticas de seguridad complejas basadas en reglas.

Quizá el modelo de control de acceso que más popularidad haya alcanzado en los últimos tiempos sea el modelo de control de acceso basado roles (RBAC) (NIST, 2003). OASIS lo ha tenido en cuenta y ofrece a través de su especificación *XACML Profile por RBAC* (TC, 2004) un método para aplicar este modelo en el contexto de los servicios Web.

Ilustración 10. Aplicación de una política de seguridad XACML.

4.1.3 Interoperabilidad de los estándares de seguridad

El consorcio WS-I (www.ws-i.org) es otro consorcio abierto al que pertenecen alrededor de 150 compañías cuyo principal papel en los servicios Web es promover la interoperabilidad entre plataformas, la adopción de esta forma de computación distribuida, y acelerar su desarrollo actuando como guía y definiendo catálogos de buenas prácticas así como cualquier otro tipo de recurso que mejore su interoperabilidad. La seguridad no queda fuera del alcance de este

propósito y, como muestra, se encuentra actualmente en desarrollo la especificación WS-I Basic Security Profile 1.0 (WS-I, 2004) que define una serie de restricciones sobre el uso de los estándares de seguridad en los servicios Web. Esta especificación normaliza el uso de los estándares de seguridad descritos en este artículo, indicando qué versiones y de qué manera deben ser aplicados.

De esta forma, si disponemos de un proceso de negocio que queremos hacer accesible desde Internet, y queremos emplear los estándares de seguridad aquí mencionados, estando además seguros de que estamos realizando un uso estándar y correcto de los mismos, sería ideal ajustarnos a este perfil con el objeto de poder integrarlo con la mayor rapidez posible en el mercado. En la figura 6 se refleja que el portal www.example.com emplea las directrices de uso indicadas por el perfil WS-I Basic Security Profile 1.0:

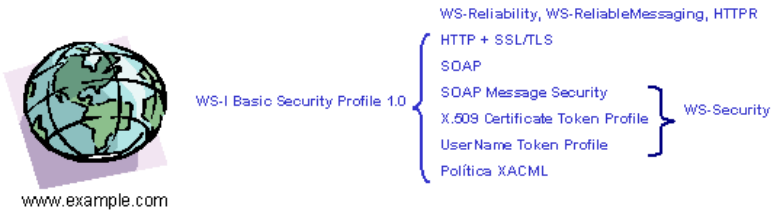

Ilustración 11. Interoperabilidad de los estándares de seguridad actuales.

4.1.4 Aspectos de la seguridad más avanzados

Los estándares mencionados en este artículo permiten contextualizar las primitivas de seguridad básicas estudiadas en el entorno de los servicios Web. Existe una gran abanico de aspectos de la seguridad en los servicios Web no mencionados como por ejemplo: la gestión de dominios de confianza y su federación (p.e.: soluciones de autenticación Single Sign-On), gestión distribuida de las políticas de seguridad a gran escala, privacidad, contextos de seguridad, integración con la seguridad a nivel de red, ampliación de la funcionalidad en los elementos empleados para seguridad del perímetro, seguridad de la Web semántica, etc. Se está realizando un gran esfuerzo por parte de la industria para elaborar especificaciones que cumplan y estandaricen la definición y aplicación de todos estos aspectos. Veremos una gran evolución en los dos próximos años.

4.1.5 Conclusiones

La conclusión fundamental a la que se debe llegar tras leer este artículo es que, tanto si disponemos de servicios de empresa abiertos al mundo Internet como si planeamos hacerlo, ya existen un conjunto de estándares a nuestro disposición, cuyo núcleo principal es WS-Security 1.0, que nos permitirán garantizar los servicios de seguridad básicos así como promoverán la interoperabilidad de nuestras soluciones facilitando y acelerando de este modo el acceso al mercado de nuestros servicios Web de empresa.

5. ESPECIFICACIONES DE SEGURIDAD DEL W3C

5. Especificaciones de Seguridad del W3C.

El consorcio W3C es el encargado del proceso de estandarización de las especificaciones de seguridad de las tecnologías XML siguientes:

- XML Digital Signature (Imamura et al., 2002).
- XML Encryption (Imamura et al., 2002).
- XML Key Management System (Imamura et al., 2002).

Estas tres especificaciones son estudiadas en profundidad más adelante en este mismo documento. A continuación, se presenta un breve resumen de cada una de ellas.

5.1 Introducción del estándar XML Digital Signature

XML Digital Signature es una recomendación W3C desde el 2002 fruto del trabajo conjunto entre el consorcio W3C y el IETF que define cómo aplicar firmas digitales sobre contenido XML y cómo representar, definiendo un esquema XML, dicha información. Las firmas digitales son un mecanismo que garantiza la integridad de la información y se muestran como una evidencia no repudiable del generador de la misma. De esta forma, una entidad no podrá negar la autoría de una hipotética transferencia bancaria firmada digitalmente que llevó a cabo a través de un servicio Web.

Las tecnologías de firma digital existen mucho antes de esta recomendación. De hecho, el estándar de RSA PKCS #7 permite firmar digitalmente cualquier tipo de contenido, incluido por supuesto contenido XML. Sin embargo, lo que no era posible hasta que se creó esta

especificación es ofrecer una representación basada en XML de una firma digital. Tampoco era posible hasta ahora firmar parcialmente un documento XML cosa que la especificación en cuestión sí define. Además, también existían hasta ahora mecanismos que permitían asociar firmas digitales con otros elementos. Por ejemplo, la especificación Secure Multipurpose Internet Mail Extensions (S/MIME), ofrece un mecanismo para asociar una firma digital con un correo electrónico.

En XML Digital Signature una firma digital puede ser aplicada a cualquier contenido digital incluyendo XML. Una firma digital puede ser aplicada al contenido de uno o más recursos. Existen tres tipos de firmas: envolventes, envueltas y hermanas. Las dos primeras se aplican sobre información contenida dentro del mismo documento XML que transporta la firma digital. En las firmas hermanas el objeto de datos firmado y el elemento que representa la firma son hijos de un mismo padre de forma que no poseen relaciones padre-hijo entre sí.

Esta especificación define los procesos de creación de las firmas y su verificación. Al igual que ocurre con la especificación XML Encryption, esta especificación es independiente de la tecnología que la quiera utilizar de forma que se necesitan mecanismos adicionales que estandaricen su uso en el intercambio de mensajes entre servicios Web.

5.2 Introducción del estándar W3C XML Encryption

W3C XML Encryption es una propuesta de recomendación desde el año 2002 y ofrece un modelo de procesamiento para cifrar, descifrar y representar, en XML, los siguientes tipos de información:

- Documentos XML completos.
- Elementos XML únicos (y todos sus descendientes) dentro de un documento XML.
- El contenido de un elemento XML (algunos o todos los nodos hijos incluyendo todos sus descendientes) localizado dentro de un documento XML.
- Contenidos binarios arbitrarios ubicados fuera de un documento XML.

XML Encryption es la solución al problema de la confidencialidad de los mensajes SOAP intercambiados entre los servicios Web. XML Encryption, además de describir la estructura y la sintaxis de los elementos XML necesarios para presentar información cifrada, especifica los pasos que se deben seguir a la hora de cifrar y descifrar documentos XML (o partes de ellos).

La filosofía a la hora de aplicar el cifrado sobre la totalidad o parte de un documento XML es que las partes cifradas son substituidas por elementos definidos en la especificación XML Encryption que transportan el cifrado. De la misma manera la información cifrada será convertida en la información original cuando se realice el descifrado.

Los servicios Web utilizan XML para transportar la meta-información necesaria (en forma de cabeceras SOAP), y para transportar la carga útil o información de negocio. Por tanto, pueden aprovechar esta primitiva de seguridad para cifrar/descifrar aquellas partes de los mensajes que crean oportunas. XML Encryption no define específicamente cómo cifrar los mensajes intercambiados entre los servicios Web, dejando esta tarea a especificaciones de más alto nivel que definen las reglas de uso de esta primitiva en el contexto del intercambio de información XML mediante mensajes SOAP. Esta especificación también describe el método a utilizar para cifrar contenido cifrado (super-encryption) así como el mecanismo para cifrar claves utilizadas en el proceso de cifrado.

Si nos remontamos al principio de está sección donde se listan los tipos de datos que XML Encryption puede cifrar, podemos echar en falta la posibilidad de cifrar nodos de un árbol sin tener por que cifrar sub-árboles completos. Una solución al problema es extraer los nodos a cifrar del documento original, cifrar cada uno de ellos, y recopilarlos en un pool de nodos cifrados. Al recipiente le llegará el documento con los nodos extraídos así como el pool de nodos cifrados y será capaz de descifrar y reubicar en el árbol original aquellos nodos destinados para él (Geuer-Pollmann, 2002).

Otro problema de seguridad es la declaración explícita que se realiza de las partes que han sido cifradas. Tal y como indica la especificación, la información que es cifrada es substituida por elementos XML que contienen la información cifrada. Esto puede ser un problema cuando los

datos XML se enfrentan a ataques de análisis de la información como aquel de la poli-instanciación múltiple con las bases de datos.

La recursividad es otro problema que la especificación señala pero que no resuelve: la clave cifrada A puede requerir la clave cifrada B pero ésta, a su vez, puede necesitar también la clave cifrada A. La especificación expresa que es responsabilidad de la aplicación cifradora/descifradora manejar estas situaciones correctamente. Otro punto, más organizativo que de seguridad, que se debe resolver es la dependencia establecida con la especificación XML Digital Signature para temas tan triviales como la información del material de clave utilizado para el cifrado. Actualmente la recomendación indica el uso del elemento del espacio de nombres ds: (comúnmente asociado con el espacio de nombres que contiene los elementos definidos en XML Digital Signature XML) en vez agrupar estos elementos de utilidad de seguridad en un espacio de nombres independiente como por ejemplo se define en la familia WS-Security.

5.3 Introducción del estándar XML Key Management System

XML Key Management System es una especificación sometida al proceso de estandarización del W3C que propone un formato de información así como los protocolos necesarios para convertir una infraestructura PKI en un servicio Web de forma que se pueda:

- Registrar pares de clave privada/pública.

- Localizar claves públicas.
- Validar la clave.
- Revocación de clave.
- Recuperación de clave.

De esta forma toda la infraestructura PKI se extiende al mundo XML y permite delegar las decisiones de confianza a sistemas especializados simplificando el desarrollo de las aplicaciones cliente.

XKMS se compone de dos partes fundamentales:

- X-KISS. Define los protocolos necesarios para soportar el procesamiento de la información de clave asociada con una firma digital o datos XML cifrados.
- X-KRSS. Define los protocolos necesarios para soportar el registro de un par de claves por un poseedor de pares de claves.

XKMS se presenta como la solución a la creación de un servicio de confianza que ofrezca todos los servicios subyacentes de una infraestructura PKI ya establecida.

5.4 XML Digital Signature

Organización	W3C
Estado	Recomendación W3C desde 12 de Febrero de 2002.
Versión	N.A.

5.4.1 Introducción

El núcleo fundamental de la especificación XML Digital Signature viene definida en el documento "XML-Signature Syntax and Processing" (http://www.w3.org/TR/xmldsig-core/).

Esta especificación proporciona la sintaxis y las reglas de procesamiento para crear y representar firmas digitales. Una firma digital puede ser aplicada sobre cualquier contenido digital incluyendo XML y al contenido de uno o más recursos. Existen firmas envueltas ('enveloped') y firmas envolventes ('enveloping') siendo ambas firmas aplicadas sobre **datos contenidos en el mismo documento XML** que transporta la firma digital. También existen firmas desligadas ('detach') que firman contenido digital no contenido en el documento XML que transporta la firma.

Las firmas digitales XML se aplican sobre contenido digital (denominado en la especificación como objetos de datos) arbitrario mediante una indirección o referencia. Los objetos de datos son resumidos ('digested'), es decir se aplica una función de 'hashing' sobre ellos, y el valor resultante es ubicado en un elemento del documento XML (junto con otra información) y ese elemento es a su vez resumido de nuevo y criptográficamente firmado.

Las firmas digitales XML se representan mediante el elemento *Signature* que posee la siguiente estructura (donde '¿' significa 1 ó mas ocurrencias y '* ' significa 0 ó más ocurrencias):

```
<Signature ID?>
        <SignedInfo>
                <CanonicalizationMethod/>
                <SignatureMethod/>
                (<Reference URI? >
                        (<Transforms>)?
                        <DigestMethod>
                        <DigestValue>
                </Reference>)+
        </SignedInfo>
        <SignatureValue/>
                (<KeyInfo>)?
                (<Object ID?>)*
</Signature>
```

Las firmas digitales están asociadas a los objetos de datos vía URIs incluidas como atributos en los elementos *Reference*. Dentro de un documento XML, las firmas están relacionadas a los objetos de datos

locales mediante identificadores de fragmentos. Este tipo de datos locales puede ser incluido dentro de una *firma XML envolvente* o puede encerrar una *firma XML envuelta.* Las firmas XML desligadas se realizan sobre recursos de red u objetos de datos locales que, cuando residen dentro del mismo documento que porta la firma, sólo pueden existir como elementos 'hermanos'. En este caso la firma XML no es ni envolvente (porque la firma es un elemento 'padre') ni es envuelta (porque en este caso la firma es un elemento 'hijo').

5.4.2 Ejemplo básico

El siguiente ejemplo muestra una firma digital XML 'detached' o desligada:

```
<Signature Id="MiPrimeraFirma"
      xmlns="http://www.w3.org/2000/09/xmldsig#">

   <SignedInfo>

      <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-
           xml-c14n-20010315" />

      <SignatureMethod
     Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1" />

      <Reference URI="http://www.w3.org/TR/2000/REC-xhtml1-
20000126/">
         <Transforms>
            <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-
                 c14n-20010315" />
          </Transforms>

         <DigestMethod
               Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />

         <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>

      </Reference>

    </SignedInfo>
```

```
<SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>

<KeyInfo>
  <KeyValue>
    <DSAKeyValue>
      <P>...</P>
      <Q>...</Q>
      <G>...</G>
      <Y>...</Y>
    </DSAKeyValue>
  </KeyValue>
</KeyInfo>
</Signature>
```

El elemento requerido *SignedInfo* representa la información que se está firmando realmente. De hecho, el elemento que es normalizado, según el algoritmo definido en el elemento *CanonicalizationMethod* y que es firmado es el elemento *SignedInfo.* La validación fundamental de la firma consiste en **dos pasos obligatorios**:

- **Validación de la firma**: realizada sobre el resultado de firmar el elemento *SignedInfo* y que es transportado como valor textual del elemento *SignatureValue*.
- **Validación de la referencia**: validación de cada elemento 'digest' incluido en cada elemento *Reference*, incluido a su vez dentro del elemento *SignedInfo.* Esta validación comprueba la integridad de los objetos de datos que han sido firmados.

El elemento *CanonicalizationMethod* refleja el algoritmo utilizado para normalizar o normalizar el elemento *SignedInfo* antes de calcular su valor de digestión como parte de la operación de cálculo de la firma.

Por su parte, el elemento *SignatureMethod* indica el algoritmo utilizado para generar la firma, es decir, indica la operación utilizada para convertir el 'digest' de la forma normalizada del elemento *SignedInfo* en el valor expuesto por el elemento *SignatureValue.*

El nombre del algoritmo se firma para resistir ataques basados en sustitución por algoritmos más débiles. La especificación define una serie de algoritmos que cualquier implementación conforme a la especificación debe incluir.

Cada elemento *Reference* incluye el algoritmo 'digest' y el valor resultante de aplicarlo sobre el dato referenciado. También podría incluir algoritmos de transformaciones que produjeron la entrada para la función resumen. Un objeto de datos es firmado calculando el valor de su 'digest' y firmando ese valor. La firma es más tarde verificada mediante la referencia a los objetos de datos y el **proceso de validación de firma**.

El elemento *KeyInfo* indica la clave (o información que permite al receptor del mensaje conocerla) que debe ser utilizada para validar la firma. Posibles identificaciones incluyen certificados, nombres de claves y algoritmos de acuerdo de clave. Este elemento es opcional.

Revisemos un poco más detenidamente el elemento *Reference*.

```
<Reference URI="http://www.w3.org/TR/2000/REC-xhtml1-
20000126/">
```

```
<Transforms>
    <Transform
        Algorithm="http://www.w3.org/TR/2001/RE
        C-xml-c14n-20010315" />
 </Transforms>

<DigestMethod
      Algorithm="http://www.w3.org/2000/09/xmldsig
      #sha1" />

<DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</Diges
      tValue>

</Reference>
```

El atributo URI es opcional e identifica el objeto de datos que debe ser firmado. Este atributo se podría omitir a lo sumo en un elemento *Reference* dentro del mismo elemento *Signature*. Si se omite el atributo URI la aplicación receptora debe conocer la identidad del objeto que está siendo firmado. Por ejemplo, un protocolo de datos ligero podría omitir este atributo dado que la identidad del objeto es parte del contexto de la aplicación. Este atributo podría ser omitido de como mucho un elemento *Reference* dentro del mismo elemento *SignedInfo* o *Manifest* (veremos más adelante el significado de este elemento).

El elemento *Tranforms* es opcional y contiene el conjunto de transformaciones aplicadas sobre el objeto de datos (al contenido del recurso) antes de calcular su 'digest'. Entre las posibles transformaciones están: normalización, codificación/decodificación, XSLT, XPath, validación del esquema XML.

El elemento *DigestMethod* indica el algoritmo aplicado a los datos después de haberles aplicado las transformaciones y que produce el valor contenido en el elemento *DigestValue*. La firma del elemento *DigestValue* es lo que vincula a los recursos u objetos de datos con la clave del firmante.

5.4.3 Ejemplo extendido (I)

La especificación XML Digital Signature no especifica mecanismos para realizar declaraciones o afirmaciones asociadas con la firma y permitir, así, su firmado. En vez de esto, define lo que significa que algo esté firmado (integridad, autenticidad del mensaje y autenticación del firmante). Aquellas aplicaciones que deseen representar otras semánticas deben utilizar otras tecnologías como XML, Resource Description Framework (RDF) o WS-Security.

Estas declaraciones pueden ser firmadas incluyendo un elemento *Reference* dentro de *SignedInfo* que apunte a un elemento *SignatureProperties*. Mientras que la aplicación firmante debería ser muy cuidadosa sobre lo que firma (debería entender que es lo que contiene el elemento *SignatureProperties*) la aplicación receptora no tiene la obligación de entender esa semántica. Cualquier contenido sobre la generación de la firma podría esta ubicado dentro de un elemento *SignatureProperty.* El atributo obligatorio *Target* referencia el elemento *Signature* al cual la propiedad aplica.

Consideremos el ejemplo anterior con una referencia adicional a un elemento *Object* local que incluye un elemento *SignatureProperty*. De esta forma la firma sería no sólo de tipo desligada sino también envolvente (ya que el elemento *Signature* envuelve datos firmados).

```
<Signature Id="MiSegundaFirma" ...>

      <SignedInfo>
        ...
         <Reference URI="http://www.w3.org/TR/xml-
           stylesheet/" />
           ...
         <Reference URI="#propiedadesFirma"
           Type="http://www.w3.org/2000/09/xmldsig#Signatur
           eProperties">
          <DigestMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig
              #sha1" />

              <DigestValue>k3453rvEPO0vKtMup4NbeVu8nk=</
              DigestValue>
       </Reference>
    </SignedInfo>
  ...
  <Object>
     <SignatureProperties>
        <SignatureProperty Id="propiedadesFirma"
             Target="#MiSegundaFirma">
          <timestamp
               xmlns="http://www.ietf.org/rfcXXXX.txt">
            <date>19990908</date>
            <time>14:34:34:34</time>
          </timestamp>
        </SignatureProperty>
      </SignatureProperties>
    </Object>
</Signature>
```

La estructura simplificada del documento anterior podría ser vista de la siguiente manera:

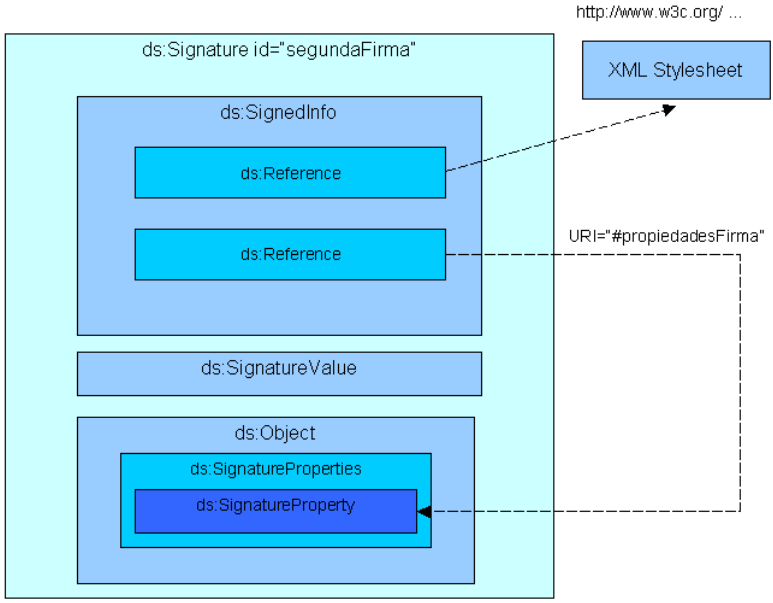

Ilustración 12. Estructura un documento XML con firma digital.

El atributo opcional *Type* del elemento *Reference* proporciona información sobre el tipo de recurso identificado por la URI. En particular, puede indicar que es un elemento *Object*, *SignatureProperty*, o *Manifest*. Este atributo puede ser utilizado por las aplicaciones para iniciar un procesamiento especial de algunos de los elementos *Reference*.

El elemento *Object* es opcional y sirve para incluir objetos de datos dentro del propio elemento de firma o en cualquier otro lugar. Las propiedades de la firma, como el instante de tiempo de generación de la firma, pueden ser firmadas opcionalmente siendo referenciadas desde

dentro de un elemento *Reference*. Estas propiedades son denominadas tradicionalmente “atributos” de la firma aunque ese término no tiene relación con el término de “atributo” XML.

5.4.4 Ejemplo extendido (II)

El elemento *Manifest* se proporciona para cubrir requisitos adicionales que no están resueltos directamente por las partes obligatorias de la especificación. A continuación se presentan dos requisitos y la manera en la que el elemento *Manifest* las resuelve.

En primer lugar, las aplicaciones frecuentemente necesitan firmar eficientemente múltiples objetos de datos incluso cuando la propia operación de firmado es “cara” como por ejemplo la firma basada en la clave pública. Este requisito puede ser resuelto incluyendo múltiples elementos *Reference* dentro del elemento *SignedInfo* ya que la inclusión de cada ‘digest’ asegura la integridad y autenticidad de los datos digeridos. Sin embargo, algunas aplicaciones podrían no querer adoptar el comportamiento definido por la validación fundamental asociada con esta aproximación porque requiere que cada elemento *Reference* incluido dentro del elemento *SignedInfo* se “someta” a la validación de la referencia (donde los valores de los elementos *DigestValue* son verificados). Estas aplicaciones podrían desear reservar la lógica de la validación de la referencia a sí mismas omitiéndola o aplicándola cuando ellas quisieran. Por ejemplo, una aplicación podría recibir un elemento *SignedInfo* que incluye elementos *Reference*. Según el proceso de validación fundamental de la especificación, basta con que una sola de

las verificaciones sobre las referencias falle para que el proceso de validación de firma deba darse por fallido en su totalidad. Sin embargo la aplicación podría desear tratar la firma pese a que las validaciones de las referencias fallaran. Para conseguir esto, el elemento *SignedInfo* debe referenciar un elemento *Manifest* que contiene uno o más elementos *Reference* (con la misma estructura que los contenidos en el elemento *SignedInfo*). De esta forma, la validación de las referencias del elemento *Manifest* estará bajo control de la aplicación.

Segundo, consideremos una aplicación donde se aplican muchas firmas (utilizando claves distintas) a un gran número de objetos de datos. Una solución ineficiente es tener una firma separada (por clave) repetidamente aplicada a un elemento *SignedInfo* muy grande (con muchos elemento *Reference*); esto sería excesivo y redundante. Una solución más eficiente es incluir muchas referencias en un único elemento *Manifest* que sea referenciado desde múltiples elementos *Signature*.

El ejemplo que se muestra a continuación incluye un elemento *Reference* que firma un elemento *Manifest* ubicado dentro de un elemento *Object*.

```
<Reference
      URI="#MiPrimerManifiesto"
      Type="http://www.w3.org/2000/09/xmldsig#Manifest">

  <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
  <DigestValue>345x3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
</Reference>
```

```
   ...
<Object>
  <Manifest Id="MiPrimerManifiesto">
    <Reference>...</Reference>
    <Reference>...</Reference>
  </Manifest>
</Object>
```

5.4.5 Reglas de procesamiento

En este apartado describimos las operaciones que forman parte de los procesos de generación y validación de una firma.

El proceso de generación fundamental de la firma incluye como pasos obligatorios la generación de los elementos *Reference* y del elemento *SignatureValue* a partir de aplicar el algoritmo de normalización del elemento *SignedInfo.*

La generación de los elementos *Reference* consiste en los siguientes pasos aplicados por cada objeto de datos que debe ser firmado:

1. Aplicar las transformaciones indicadas en el elemento *Tranforms* a los objetos de datos.

   Info –> T(Info)

2. Calcular el valor de la función resumen sobre el objeto de datos resultados:

T(Info) –> Digest(T(Info)) –> D(T(Info))

3. Crear un elemento *Reference* incluyendo la identificación del objeto de datos (opcional), cualquier transformación aplicada (opcional), el algoritmo de la función de resumen y el valor resultante (*D(T(Info)*) dentro de un elemento *DigestValue*.

5.4.6 Generación de la firma

Por su parte el proceso de generación de la firma se compone de los siguientes pasos:

1. Crear un elemento *SignedInfo* que incluya los elementos *SignatureMethod*, *CanonicalizationMethod* y uno o varios elementos *Reference*.
2. Normalizar el elemento *SignedInfo* y después calcular el valor del elemento *SignatureValue* a partir de los algoritmos especificados en el elemento *SignedInfo.*

5.4.7 Validación de la firma

Validación Fundamental

Los pasos obligatorios del proceso de *validación fundamental* incluye el proceso ***validación de las referencias*** *(*verificación del resumen

contenido en cada elemento *Reference* del elemento *SignedInfo)* y la ***validación de la firma*** calculada sobre el elemento *SignedInfo.*

Destacar que podría haber firmas válidas que algunas aplicaciones de firmado fueran incapaces de validar. Razones por las que esto se pueda dar son la posible implementación de las partes opcionales que define la especificación (ver ejemplos extendidos), incapacidad o no disposición a ejecutar ciertos algoritmos, o incapacidad o no disposición a deshacer la referencia a ciertas URIs (algunos esquemas de URIs podrían causa efectos no deseados), etc.

Validación de las Referencias

1. Normalizar el elemento *SignedInfo* en función del algoritmo indicado en el elemento *CanonicalizationMethod* del elemento *SignedInfo.*
2. Para cada elemento *Reference* dentro del elemento *SignedInfo*:

- Obtener el objeto de datos del que debe calcularse el resumen.
- Aplicar la función resumen sobre el objeto de datos resultante utilizando el algoritmo definido en el elemento *DigestMethod* de su correspondiente elemento *Reference*.
- Comparar el valor del resumen generado contra el valor del elemento *DigestValue* contenido dentro del elemento *Reference*. Si no coinciden la validación debe darse por fallida en su totalidad.

Es importante hacer notar que en el paso 1 el elemento *SignedInfo* es normalizado. La aplicación debe asegurar que el método de normalización definido en el elemento *CanonicalizationMethod* no posee efectos secundarios peligrosos como por ejemplo la reescritura de URIs.

Validación de la Firma

1. Obtener la información relacionada con las claves que intervienen del elemento *KeyInfo* o de una fuente externa. Recordar que el elemento *KeyInfo* es opcional y sólo tiene como obligación proporcionar información suficiente para que cierto receptor del mensaje sepa qué clave utilizar para validar la firma.
2. Obtener la forma canónica del elemento *SignatureMethod* utilizando el algoritmo especificado en el elemento *CanonicalizationMethod* y utilizar el resultado (junto con la información de la(s) clave(s) obtenida anteriormente) para confirmar el valor del elemento *SignatureValue* sobre el elemento *SignedInfo.*

5.4.8 Algoritmos

Los algoritmos son identificados por URIs que aparecen como un atributo del elemento que identifica el papel del algoritmo (*DigestMethod, Transform, SignatureMethod,* o *CanonicalizationMethod*). Todos los algoritmos utilizados toman parámetros pero en muchos de los caso los parámetros pueden ser implícitos.

Por ejemplo, a un elemento *SignatureMethod* se le pasa implícitamente dos parámetros: la información de las claves y la salida del elemento *CanonicalizationMethod*. La especificación XML Digital Signature define un conjunto de algoritmos, sus URIs y requisitos para su implementación.

Los requisitos se especifican sobre la implementación, no sobre los requisitos para su uso en firmas. Es más, el mecanismo es extensible; se podrían utilizar algoritmos alternativos.

5.4.9 Normalización XML y consideraciones de restricciones de la sintaxis

El más ligero cambio a un mensaje sobre el que se están aplicando operaciones criptográficas resultará en un valor completamente diferente. Esto proporciona la seguridad de la integridad del mensaje pero introduce una complicación extra: dos documentos podrían diferir en una comparación textual aunque fueran lógicamente equivalentes. Cosas como delimitadores, etiquetas vacías, uso de valores hexadecimales en vez de nombres en atributos o variaciones en los comentarios son puntos que pueden diferir entre dos documentos que estructuralmente son iguales. La especificación ***Canonicalización XML*** (estudiado en este mismo documento) describe un método para generar una representación física de un documento, conocido como la forma canónica, de tal forma que si dos documentos tienen la misma forma canónica entonces los dos

documentos deben ser considerados lógicamente equivalentes dentro de un contexto de aplicación dado.

Esto es particularmente importante para la creación de firmas digitales XML porque es claramente necesario que variantes textuales que no tienen implicaciones lógicas no debieran sugerir que la integridad de un documento o la autenticación de su emisor sean sospechosas. Esto es algo que por otro lado podría ocurrir como resultado de un tratamiento distinto por herramientas diferentes generando textos diferentes y, a su vez, distintos resúmenes de mensajes. Por lo tanto, durante la generación de la firma y el proceso de validación del mismo resumen de los mensajes deben ser calculados sobre sus formas canónicas. Si los resúmenes coinciden, entonces esto confirma que las formas canónicas sobre las cuales son calculados también coinciden, incluso aunque la representación textual difiera.

5.4.10 Consideraciones de seguridad

La especificación XML Signature proporciona un mecanismo muy flexible de firma digital pero se deben tener en cuenta una serie de aspectos concernientes a la seguridad.

Transformaciones

Un requisito de esta especificación es permitir que las firmas sean aplicadas sobre una parte o la totalidad de un documento XML. Los mecanismos de transformación cumplen este requisito permitiéndole a uno firmar datos derivados de procesar contenidos del recurso

identificado. Recordemos que en esta especificación se pueden incluir en dos puntos semánticas de transformación:

- Como método de normalización aplicado sobre el elemento *SignedInfo.*
- Transformaciones opcionales incluidas como primer elemento hijo de los elementos *Reference* y que aplican las transformaciones indicadas sobre los objetos de datos justo antes de calcular su función resumen.

Por ejemplo, aquellas aplicaciones que deseen firmar un formulario, pero que permiten a los usuarios introducir datos limitados en un campo sin invalidar una firma digital anterior sobre el formulario podría utilizar XPath para excluir esas porciones que el usuario necesita cambiar. Se podría especificar arbitrariamente un elemento *Transforms* que incluyera las transformaciones de codificación, instrucciones de normalización o incluso transformaciones XSLT. A continuación se muestran tres advertencias causadas por esta característica:

Solamente lo que esté firmado es seguro

En primer lugar, obviamente, las firmas sobre un documento transformado no "aseguran" la información descartada por la transformación: **solamente lo que es firmado es seguro.**

Destacar que el uso de la forma canónica inclusiva XML asegura que todas las entidades internas y los espacios de nombre XML son

expandidos dentro del contenido que es firmado. Todas las entidades son reemplazadas por sus definiciones y la forma canónica explícitamente representa el espacio de nombres que de otra forma un elemento heredaría. Las aplicaciones que no normalizan el contenido XML (especialmente el elemento *SignedInfo*) no deberían utilizar entidades internas y deberían representar el espacio de nombres explícitamente dentro del contenido que está siendo firmado ya que no pueden confiar en que la normalización lo haga por ellos. Los usuarios preocupados con la integridad de las definiciones de los tipos de elementos asociados con la instancia XML que es firmada también podrían desear firmar esas definiciones (p.e.: el esquema XML, DTD).

En segundo lugar, un sobre SOAP que contiene información firmada no está "asegurado" por la firma. Por ejemplo, cuando un sobre cifrado contiene una firma, la firma no protege la autenticidad o integridad de las cabeceras del sobre que no han sido firmadas ni su forma textual cifrada, solamente "asegura" el texto plano que realmente ha sido firmado.

Solamente lo que es "visto" debería ser firmado

Adicionalmente, la firma "asegura" cualquier información introducida por la transformación: solamente lo que es "visto" (lo cual es representado al usuario visualmente, por audio u otros medios), debería ser firmado.

"Ver" lo que es firmado

De la misma manera en la que un usuario solamente debería firmar lo que él o ella "ve", personas o máquinas automatizadas que confían en la validez de un documento transformado porque disponen de una firma válida deberían operar sobre los datos que fueron transformados(incluyendo la normalización) y firmados, y no sobre los datos originales pretransformados. Esta recomendación aplica tanto a las transformaciones reflejadas en la firma como aquellas incluidas como parte del documento mismo. Por ejemplo, si un documento XML incluye una hoja de estilos XSLT, es el documento transformado el que debería ser representado al usuario y el que debería ser firmado. Para cumplir esta recomendación en la que un documento referencia una hoja de estilos externa, el contenido de ese recurso externo debería ser firmado también mediante un elemento *Signature*, de otra forma el contenido de ese recurso externo podría variar lo cual alteraría el documento resultante sin tener que invalidar la firma (lo que debería ocurrir).

Algunas aplicaciones podrían operar sobre los datos originales, o alguna de sus formas intermedias, pero deberían ser extremadamente cuidadosas sobre la debilidad potencial introducida entre los datos originales y los transformados. Consideremos un algoritmo de normalización que normaliza la tipografía de los caracteres (de minúsculas a mayúsculas) o su composición ('e y acento' para 'e acentuada'). Un adversario podría introducir cambios que son normalizados y consecuentemente intranscendentes para la validez de la firma pero palpables para un procesador DOM. Por ejemplo, modificando la tipografía de un carácter

se podría influenciar el resultado de una petición XPath. Un riesgo muy serio aparece si ese cambio es normalizado para la validación de una firma pero el procesador opera sobre los datos originales y devuelve un resultado diferente del intencionado. En consecuencia:

- Todos los documentos sobre los que se opera y generados por aplicaciones de firmado deben estar en formato NFC o NFC-Corrigendum. De otra forma un procesador intermedio podría involuntariamente romper la firma.
- No deberían ser hechas codificaciones de normalizaciones como parte de una transformación de una firma o si ocurre la normalización, la aplicación debería ver (poder operar sobre) la forma normalizada.

Verificar el modelo de seguridad

Esta especificación utiliza firmas de clave pública y códigos hash de autenticación con clave. Con claves públicas cualquier número de partes puede disponer de la clave pública y verificar firmas mientras que solamente las partes que poseen las claves privadas pueden generar firmas. El número de partes que poseen la clave privada debería ser minimizado y preferiblemente sólo uno. Un punto importante es la seguridad de los verificadores en la clave pública que utilizan y su vinculación con la entidad o capacidades representadas por la correspondiente clave privada, normalmente solucionado mediante certificados y sistemas de autoridad online. Normalmente los códigos hash de autenticación, basados en claves secretas, son mucho más eficientes en términos de esfuerzo computacional pero tiene la

característica de que todos los verificadores deben poseer la misma clave secreta que el firmante. Por lo tanto cualquier verificador tiene la capacidad potencial de falsificar la firma.

La especificación permite algoritmos de firma proporcionados por el usuario. Tales algoritmos podrían tener diferentes modelos de seguridad. Por ejemplo, los métodos implicados en biometría normalmente dependen de características físicas del usuario autorizado que no pueden ser cambiadas de la misma manera en la que las claves públicas o secretas pueden serlo.

Algoritmos, Longitud de Claves, Certificados, Etc.

La fortaleza de una firma particular depende de todos los eslabones en la cadena de seguridad. Esto incluye los algoritmos de resumen y de firma utilizados, la fortaleza de la generación de claves y el tamaño de la misma, la seguridad de la clave y la autenticación del certificado y los mecanismos de distribución, la política de validación de la cadena de certificación, la protección del proceso criptográfico de observación hostil.

Las aplicaciones deben tener mucho cuidado en la ejecución de los diversos algoritmos implicados en la creación de una firma digital y en el procesamiento de cualquier “contenido ejecutable” que pudiera ser pasado como parámetros a dichos algoritmos, como por ejemplo transformaciones XSLT. Los algoritmos definidos por la especificación deben ser implementados mediante una librería de confianza. Se debe

tener más cuidado incluso con algoritmos definidos por las propias aplicaciones.

La seguridad de un sistema global dependerá también de la seguridad e integridad de sus procedimientos operativos, su personal, y el uso administrativo de esos procedimientos. Todos los factores listados en esta sección son importantes para la seguridad global de un sistema; sin embargo están más allá del alcance de la especificación XML Digital Signature.

5.5 XML Encryption

Organización	W3C
Estado	Liberada.
Versión	1.2

5.5.1 Introducción

La especificación define un proceso para cifrar datos y representar el resultado en XML. Los datos podrían ser arbitrarios (incluyendo un documento XML), un elemento XML, o el contenido de un elemento XML.

El resultado de cifrar un elemento es un elemento XML *EncryptedData* el cual contiene o identifica (mediante una URI) los datos cifrados.

Un documento XML, o el contenido de un elemento XML que es cifrado, es reemplazado por su correspondiente elemento *EncryptedData* en el documento XML cifrado de salida. Como se cifran datos arbitrarios, el elemento *EncryptedData* podría convertirse en un elemento raíz del nuevo documento XML de salida cuyas partes han sido cifradas.

5.5.2 Elemento EncryptedData

Esquemáticamente, el elemento *EncryptedData* presenta la siguiente estructura (donde "?" denota 0 o una ocurrencia, "+" denota una o más ocurrencias, "*" denota cero o más ocurrencias y una marca de elemento vacío significa que el elemento debe estar vacío):

```
<EncryptedData Id? Type? MimeType? Encoding?>
        <EncryptionMethod/>?
        <ds:KeyInfo>
                <EncryptedKey>?
                <AgreementMethod>?
                <ds:KeyName>?
                <ds:RetrievalMethod>?
                <ds:*>?
        </ds:KeyInfo>?
        <CipherData>
                <CipherValue>?
                <CipherReference URI?>?
        </CipherData>
        <EncryptionProperties>?
</EncryptedData>
```

El elemento *CipherData* puede "envolver" o referenciar los datos cifrados. Si envuelve directamente los datos cifrados éstos se representan como el contenido del elemento *CipherValue;* por otro lado, si utiliza el atributo URI del elemento *CipherReference* entonces apunta a la ubicación de los datos cifrados.

En resumen, cuando se realiza el cifrado de datos de un 'infoset' XML, se sustituye la parte cifrada por un elemento *EncryptedData* definido por la especificación en cuestión. Este elemento permite:

- Definir el algoritmo de cifrado utilizado.
- Especificar la clave de cifrado utilizada.
- Contener o referenciar los datos cifrados.
- Contemplar atributos específicos de los datos cifrados (meta-información).

Por último señalar que es absolutamente factible cifrar cualquier tipo de elemento MIME.

5.5.3 Granularidad del cifrado. Ejemplos.

Consideremos la siguiente información ficticia que incluye información de identificación e información necesaria para cierto sistema de pago (p.e.: tarjeta de crédito, transferencia de dinero, o cheques electrónicos).

```
<?xml version="1.0" ?>
  <PaymentInfo xmlns="http://example.org/paymentv2">
    <Name>John Smith</Name>
    <CreditCard Limit="5,000" Currency="USD">
       <Number>4019 2445 0277 5567</Number>
       <Issuer>Example Bank</Issuer>
       <Expiration>04/02</Expiration>
    </CreditCard>
</PaymentInfo>
```

Este documento representa que John Smith está utilizando su tarjeta de crédito la cual posee un límite de 5000 dólares.

Obviamente la información del número de tarjeta de John Smith es muy delicada y debería ser debidamente protegida contra posibles ataques. Si quisiéramos que nuestra aplicación mantenga esta información de manera confidencial, podríamos cifrarla en un elemento *CreditCard*:

```
<?xml version="1.0" ?>
<PaymentInfo xmlns="http://example.org/paymentv2">
    <Name>John Smith</Name>
    <EncryptedData
        Type="http://www.w3.org/2001/04/xmlenc#Eleme
        nt" xmlns="http://www.w3.org/2001/04/xmlenc#">
        <CipherData>
            <CipherValue>A23B45C56</CipherValue>
        </CipherData>
    </EncryptedData>
</PaymentInfo>
```

Cifrando el elemento *CreditCard* íntegramente desde su marca de comienzo hasta su marca final, la propiedad identidad del elemento queda escondida. Un "fisgón" no puede saber si se está llevando a cabo una transferencia o un pago por tarjeta de crédito. El elemento *CipherData* contiene la serialización cifrada del elemento *CreditCard*. Como también puede observarse, el elemento *EncryptedData* ha sustituido el elemento *CreditCard* ya que es éste el que ha sido cifrado.

Cifrando el contenido de un elemento XML

Imaginemos ahora que pudiera ser útil para ciertos agentes intermediarios conocer que John Smith utilizó una tarjeta de crédito con un límite en

particular. En este caso, se debería dejar intacto el elemento *CreditCard* cifrando sólo su contenido:

```
<?xml version="1.0" ?>
<PaymentInfo xmlns="http://example.org/paymentv2">
   <Name>John Smith</Name>
   <CreditCard Limit="5,000" Currency="USD">
      <EncryptedData  xmlns="http://www.w3.org/2001/04/xmlenc#"
            Type="http://www.w3.org/2001/04/xmlenc#Content">
         <CipherData>
            <CipherValue>A23B45C56</CipherValue>
         </CipherData>
      </EncryptedData>
   </CreditCard>
</PaymentInfo>
```

Cifrando el contenido de un elemento XML (Datos Carácter - CDATA)

Consideremos ahora el escenario en el que toda la información excepto el número de la tarjeta de crédito pueda ir sin cifrar, aunque se esté mostrando el hecho de que existen números al mantener el elemento *Number*. En este caso el documento XML cifrado tendría el siguiente aspecto:

```
<?xml version="1.0" ?>
<PaymentInfo xmlns="http://example.org/paymentv2">
  <Name>John Smith</Name>
   <CreditCard Limit="5,000" Currency="USD">
      <Number>
         <EncryptedData
            xmlns="http://www.w3.org/2001/04/xmlenc#"
```

```
                    Type="http://www.w3.org/2001/04/xmlenc#Content"
                    >
                    <CipherData>
                    <CipherValue>A23B45C56</CipherValue>
                    </CipherData>
                </EncryptedData>
            </Number>
            <Issuer>Example Bank</Issuer>
            <Expiration>04/02</Expiration>
        </CreditCard>
    </PaymentInfo>
```

Tanto el elemento *CreditCard* como el elemento *Number* se muestran en claro, aunque el contenido de datos de caracteres de éste último se encuentra cifrado.

Cifrando datos arbitrario y documentos XML

Si el escenario de la aplicación requiere que toda la información se encuentre cifrada, entonces el documento entero se cifra como una secuencia de octetos. Esto aplica a datos arbitrarios incluyendo documentos XML:

```
<?xml version="1.0" ?>
<EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
      MimeType="text/xml">
   <CipherData>
      <CipherValue>A23B45C56</CipherValue>
   </CipherData>
</EncryptedData>
```

5.5.4 Super-cifrado: cifrar el elemento EncryptedData

Un documento XML podría contener cero o más elementos *EncryptedData.* Este elemento no puede ser padre o hijo de otro elemento *EncryptedData.* Sin embargo, los datos reales cifrados pueden ser cualquier cosa, incluyendo elementos *EncryptedData* y elementos *EncryptedKey* (ambos comúnmente denominados bajo el nombre de super-cifrado). Durante el super-cifrado de alguno de estos dos elementos se debe cifrar el elemento completo. Cifrar sólo el contenido de estos elementos, o cifrar selectivamente ciertos elementos hijos provoca una instancia inválida del esquema XML.

Por ejemplo, consideremos el siguiente documento:

```
<pay:PaymentInfo xmlns:pay="http://example.org/paymentv2">
      <EncryptedData Id="ED1"
      xmlns="http://www.w3.org/2001/04/xmlenc#"
      Type="http://www.w3.org/2001/04/xmlenc#Element">
         <CipherData>
               <CipherValue>originalEncryptedData</CipherValue>
        </CipherData>
    </EncryptedData>
 </pay:PaymentInfo>
```

Un super-cifrado válido del elemento *EncryptedData* sería:

```
<pay:PaymentInfo xmlns:pay="http://example.org/paymentv2">
        <EncryptedData Id="ED2"
        xmlns="http://www.w3.org/2001/04/xmlenc#"
        Type="http://www.w3.org/2001/04/xmlenc#Element">
              <CipherData>
```

```
                <CipherValue>newEncryptedData</CipherValue>
            </CipherData>
        </EncryptedData>
</pay:PaymentInfo>
```

Donde el contenido del elemento *CipherValue* 'newEncryptedData' es la codificación en base 64 de la secuencia de octetos cifrada resultante de cifrar el elemento *EncryptedData* con *Id='ED1'*.

El siguiente fragmento muestra un ejemplo de uso del elemento *EncryptedData* con una clave simétrica ("claveA"):

```
<EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
      Type="http://www.w3.org/2001/04/xmlenc#Element">
    <EncryptionMethod
        Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            s:KeyName>Clave A</ds:KeyName>
        </ds:KeyInfo>
        <CipherData>
            <CipherValue>DEADBEEF</CipherValue>
        </CipherData>
</EncryptedData>
```

El tipo de los datos cifrados podría ser representado como el valor de un atributo de forma que esta información pueda ser de ayuda para el posterior descifrado y procesamiento. En este caso, el dato cifrado es un 'elemento'. Otras alternativas válidas hubieran sido el 'contenido' de un elemento, o una secuencia de octetos externa que puede también ser identificada mediante los atributos *MimeType* y *Encoding*.

El algoritmo indicado en:

```
<EncryptionMethod
    Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
```

es 3DES-CBC que es un algoritmo simétrico de cifrado.

La clave simétrica utilizada para el cifrado tiene asociada el nombre "Clave A" y el elemento *CipherData* contiene el resultado del cifrado con el algoritmo y la clave indicada. El resultado del cifrado es una secuencia de octetos en base 64 pero podría haber sido un elemento *CipherReference* en vez de un elemento *CipherData* de forma que la información cifrada hubiera sido referenciada mediante un atributo URI del elemento *CipherReference*.

5.5.5 Cifrado de una clave

La siguiente estructura del elemento *EncryptedData* es muy similar a la anterior, excepto que esta vez la clave es referenciada mediante el elemento *RetrievalMethod*.

```
<EncryptedData Id="ED" xmlns="http://www.w3.org/2001/04/xmlenc#">
    <EncryptionMethod
        Algorithm="http://www.w3.org/2001/04/xmlenc#aes128- cbc" />
    <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <ds:RetrievalMethod URI="#EK"
          Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey" />
      <ds:KeyName>Sally Doe</ds:KeyName>
    </ds:KeyInfo>
    <CipherData>
```

```
        <CipherValue>DEADBEEF</CipherValue>
    </CipherData>
</EncryptedData>
```

El algoritmo AES-128-CBC es un algoritmo de cifrado simétrico. El elemento *RetrievalMethod* se utiliza para indicar el lugar de una clave de tipo *"EncryptedKey"*. La clave AES se encuentra en *"#EK"*. El elemento *KeyName* proporciona una vía alternativa para identificar la clave necesaria para descifrar el elemento *CipherData.* Alguno o ambos de los elemento *KeyName* o *KeyRetrievalMethod* podría ser utilizado para identificar la misma clave.

Dentro del mismo documento XML, existe una estructura *EncyptedKey* que es referenciada desde el elemento *RetrievalMethod* mediante *URI='#EK'.*

```
<EncryptedKey Id="EK" xmlns="http://www.w3.org/2001/04/xmlenc#">
    <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-
        1_5" />
    <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:KeyName>Clave A</ds:KeyName>
    </ds:KeyInfo>
    <CipherData>
        <CipherValue>xyzabc</CipherValue>
    </CipherData>
    <ReferenceList>
        <DataReference URI="#ED" />
    </ReferenceList>
    <CarriedKeyName>Sally Doe</CarriedKeyName>
</EncryptedKey>
```

El elemento *EncryptedKey* es similar al elemento *EncrytedData* excepto que los datos cifrados en el primero se corresponden siempre con una clave. El método de cifrado reflejado en el elemento *EncryptionMethod* es el algoritmo asimétrico de clave pública RSA. El elemento *KeyName* cuyo valor es "Clave A" es una propiedad de la clave necesaria para descifrar, utilizando RSA, el contenido del elemento *CipherData.* El elemento *CipherValue* del elemento *CipherData* contiene el valor cifrado de la clave protegida mediante el algoritmo RSA.

El elemento *ReferenceList* identifica los objetos cifrados (*DataReference* y *KeyReference*) con esta clave. El elemento *ReferenceList* contiene una lista de referencias a datos cifrados por la clave simétrica transportada dentro de la estructura.

Por último el elemento *CarriedKeyName* es utilizado para identificar el valor de la clave cifrada el cual podría ser referenciado por un elemento *KeyReference* dentro de un elemento *KeyInfo* (ya que el valor del atributo ID debe ser único en un documento, el elemento *CarriedKeyName* puede indicar que varias estructuras *EncryptedKey* contienen el mismo valor cifrado de una clave para distintos recipientes).

5.5.6 Reglas de Procesamiento

En este apartado relataremos las reglas de procesamiento que deben ser realizadas como parte de las operativas de cifrado y descifrado. En primer lugar, distingamos los principales roles a los que nos iremos refiriendo:

- Aplicación. La aplicación que realiza la petición de un cifrado XML proporcionando los datos y parámetros necesarios para su proceso.
- Cifrador. Una implementación de la especificación XML Encryption que posee la capacidad de cifrar los datos.
- Descifrador. Una implementación de la especificación XML Encryption con el papel de descifrador de datos.

Cifrado

Para cada elemento de datos que debe ser cifrado como un elemento *EncryptedData* o *EncryptedKey* el cifrador:

1. Selecciona el algoritmo (y parámetros) que van a ser utilizados para cifrar los datos.
2. Obtiene y (opcionalmente) representa la clave.
    1. Si la clave es identificada (mediante un nombre, una URI, o incluido en un elemento hijo), construye un elemento *KeyInfo* de la manera apropiada.
    2. Si la clave a su vez debe cifrarse, se debe construir un elemento *EncryptedKey* aplicando recursivamente este proceso de cifrado. El resultado podría ser entonces un elemento hijo del elemento *KeyInfo*, o podría existir en otro lugar pudiendo ser identificada como se indicó en el paso anterior.
3. Cifrar los datos.

1. Si los datos son un ‘elemento’ o el ‘contenido’ de un elemento, se deben obtener los octetos serializando los datos en UTF-8 tal y como se expresa en la especificación XML. La serialización podría ser realizada por el cifrador. Si el cifrador no serializa, entonces la aplicación DEBE realizar la serialización.
2. Si los datos son de algún otro tipo que no sea octetos, la aplicación debe serializarlos a octetos.
3. Cifrar los octetos utilizando el algoritmo y la clave indicada en los pasos 1 y 2.
4. A menos que el descifrador conozca implícitamente el tipo de los datos cifrados, el cifrador debería proporcionar explícitamente el tipo de representación.

La definición de este tipo ligado a un identificador define cómo obtener e interpretar los octetos en texto plano tras su descifrado. Por ejemplo, el identificador podría indicar que los datos son una instancia de alguna otra aplicación (p.e.: alguna aplicación de comprensión XML) que debe ser posteriormente procesados. O, si los datos son una simple secuencia de octetos podrían ser descritos con los atributos *MimeType* y *Encoding*. Por ejemplo, los datos podrían ser un documento XML (MimeType=”text/xml”), una secuencia de caracteres (MimeType=”text/plain”), o los datos binarios de una imagen (MimeType=”image/png”).

4. Construir la estructura del elemento *EncryptedType* (*EncryptedData* o *EncryptedKey)*: una estructura *EncryptedType* representa toda la información discutida previamente incluyendo el tipo de datos cifrados, algoritmo de cifrado, parámetros, clave, tipo de los datos cifrados, etc.
    1. Si la secuencia de octetos cifrados obtenida en el paso 3 debe ser almacenada en el elemento *CipherData* dentro de un elemento *EncryptedType*, entonces la secuencia de octetos cifrada es codificada en base64 e introducida como el contenido del elemento *CipherValue*.
    2. Si la secuencia de octetos cifrados debe ser almacenada externamente a la estructura *EncryptedType*, entonces debemos almacenar o retornar la secuencia de octetos cifrados, y representar la URI y las transformaciones (si se aplicó alguna) requeridas para que el descifrador puede recuperar la secuencia de octetos cifrados dentro de un elemento *CipherReference*.

5. Procesar el elemento *EncryptedData.*
    1. Si el atributo *Type* de los datos cifrados es un ‘elemento’ o el ‘contenido’ de un elemento, entonces el cifrador debe ser capaz de devolver el elemento *EncryptedData* a la aplicación. La aplicación podría utilizar éste como el elemento de más alto nivel en un nuevo documento XML o insertarlo dentro de otro documento XML, lo cual podría requerir su codificación.

El cifrador debería ser capaz de reemplazar el ‘elemento’ o el ‘contenido’ con el elemento *EncryptedData*, Cuando una aplicación requiere que el documento XML o el contenido sea reemplazado, debe proporcionar el contexto del documento XML además de identificar el ‘elemento’ o el ‘contenido’ que debe ser reemplazado.

2. Si el atributo *Type* de los datos cifrados no es un ‘elemento’ o el ‘contenido’ de un elemento entonces el cifrador deberá siempre devolver el elemento *EncryptedData* a la aplicación. La aplicación podría utilizar éste como el elemento de más alto nivel en un nuevo documento XML o insertarlo en otro documento XML lo cual podría requerir de nuevo una recodificación.

Descifrado

Para cada elemento *EncryptedKey* derivado que debe ser descifrado, el descifrador debe:

1. Procesar el elemento para determinar el algoritmo, parámetros y el elemento *KeyInfo* que debe ser utilizado.
2. Localizar los datos de la clave de cifrado según lo indique el elemento *KeyInfo*, el cual podría contener uno o más nodos hijos. Si los datos de clave de cifrado están cifrados, se debe localizar la correspondiente clave para descifrarlos. En caso contrario uno podría recuperar los datos de la clave de cifrado de un almacén local utilizando los atributos que hayan sido proporcionados.

3. Descifrar los datos contenidos en el elemento *CipherData.*
4. Si se encuentra un elemento hijo *CipherValue*, entonces el valor del texto asociado debe ser recuperado y decodificado en base64 para así obtener la secuencia de octetos cifrada.
5. Si se encuentra un elemento *CipherReference* como elemento hijo, se deben utilizar las URIs y las transformaciones (si las hubiera) utilizadas para recuperar la secuencia de octetos cifrada.
6. La secuencia de octetos es descifrada utilizando el algoritmo, y sus parámetros, y el valor de clave ya determinado desde los pasos 1 y 2.
7. Procesar los datos descifrados de *Type* 'elemento' o 'contenido' del elemento.
    1. La secuencia de octetos en texto claro obtenida en el paso 3 es interpretada como datos de carácter codificados en UTF-8.
    2. El descifrador debe ser capaz de devolver el valor de *Type* y de los datos de carácter XML codificados en UTF-8. No es requisito indispensable que el descifrados realice alguna validación sobre el XML serializado.
    3. El descifrador debería disponer de la capacidad de reemplazar el elemento *EncrypteData* con el 'elemento' o 'contenido' del elemento descifrado representado por los caracteres codificados en UTF-8. El descifrador no tiene como obligación realizar las validaciones sobre el resultado de la operación de reemplazo.

La aplicación proporciona el contexto del documento XML e identifica el elemento *EncryptedData* que debe ser reemplazado. Si el documento en el cual está ocurriendo el reemplazamiento no fuera UTF-8, el descifrador debe codificar los caracteres codificados en UTF-8 al código del documento.

8. Procesar los datos descifrados si no se especifica el *Type* o no es un ‘elemento’ ni el ‘contenido’ de un elemento.
    1. La secuencia de octetos en texto claro obtenida en el paso 3 debe ser devuelta a la aplicación para su posterior procesamiento junto con el valor de los atributos *Type*, *MimeType* y *Encoding* (cuando éstos son especificados). El primero es obligatorio y ayuda a la aplicación a procesar o interpretar los datos descifrados. Los otros dos, *MimeType* y *Encoding* son tan sólo informativos.

       Destacar que este paso incluye el procesamiento de los datos descifrados de un elemento *EncryptedKey*. La secuencia de octetos en texto claro representa el valor de clave y es utilizada por la aplicación para descifrar otro(s) elemento(s) *EncryptedType*.

5.6 XML Key Management System

Organización	W3C
Estado	Liberada.
Versión	2.0

5.6.1 Introducción

En los dos estándares anteriores hemos visto cómo podíamos utilizar la criptografía en XML como herramienta para garantizar la integridad y la confidencialidad de la información intercambiada.

XML Key Management System (XKMS) es una recomendación del W3C desde el 2001 cuyo principal objetivo es extender la funcionalidad de las infraestructuras de clave pública (Adams & Farrell, 1999) al mundo de los servicios Web. Es decir, proporcionar una interfaz de servicio Web a una infraestructura PKI.

5.6.2 Infraestructura de Clave Pública

Las tecnología basadas en clave pública resultan ser muy flexibles y, por tanto, adaptables. Si una entidad conoce la clave pública del receptor con el que quiere comunicarse, tan sólo tendrá que utilizarla para cifrar los mensajes que le quiera enviar. Así mismo, siendo posesor de una clave privada podrá dotar a los mensajes de integridad y no repudio.

Cuando el número de partes que desean participar en la comunicación es pequeña esta aproximación, en la que las claves públicas de los participantes son distribuidas directamente por ellos, en lugar de estar en un repositorio centralizado, es válida. Pero este escenario posee un claro problema de escalabilidad, a medida que el número de participantes es más alto la capacidad de gestionar las claves públicas de todos los participantes se hace más difícil y tediosa.

Para resolver el problema de la distribución de las claves públicas existen las denominadas infraestructuras de clave pública (PKI). Estas infraestructuras actúan como repositorios centrales de las claves públicas de los principales que actúan en un mismo dominio de confianza.

Inicialmente, la idea de una PKI consistía de un repositorio donde se publicaban, valga la redundancia, las claves públicas de la misma manera en la que se publican los números de teléfonos en un listín telefónico. Cuando una entidad A desea enviar un mensaje a una entidad B, simplemente tendrá que buscar en este directorio la clave pública de B. Una vez encontrada la puede utilizar para cifrar los mensajes que le envíe o para negociar una clave simétrica que le permita realizar las operaciones de cifrado de los mensajes mucho más rápido.

El problema de esta solución tan sencilla es que Internet, como red no confiable, no garantiza que las identidades asociadas con las claves públicas sean realmente quiénes declaran ser.

Para solucionar este problema, aparece el concepto de certificado. Un certificado vincula una clave pública con una identidad. Esta vinculación está respaldada por una entidad de certificación que se encarga de garantizar esta relación identidad – clave pública. Estos certificados donde se recogen toda la información relacionada con esta vinculación (emisor del certificado, sujeto a favor de quien está emitido, clave pública, etc.) se pueden distribuir de manera independiente al directorio de claves públicas.

A lo largo de los últimos años se han propuesto diversas soluciones PKI. El siguiente cuadro resume las propuestas más significativas:

Propuesta PKI	**Descripción**
X.509	El formato de los certificados X.509 (Housley, Ford, Polk, & Solo, 1999) se empezó a utilizar como los protocolos de la familia OSI (Open Systems Interconnected). Actualmente son la base de las propuestas de PKI más conocidas.
PKIX	Public Key Infrastructure X.509 (PKIX) comenzó como un perfil de la especificación X.509 que describía el uso de los certificados X.509 en los protocolos del IETF como SSL, S/MIME o IPSEC. Desde entonces este grupo ha definido diversas extensiones de la especificación X.509 hasta tal punto de que PKI se referencia a menudo como un modelo PKI en su propio derecho.
PGP	Pretty-Good-Privacy (Zimmerman, 1994) fue diseñado por Phil Zimmerman como respuesta a la complejidad de los procedimientos establecidos para gestionar una autoridad de certificación X.509. En este modelo, cualquier entidad puede ser una autoridad de certificación y puede emitir certificados. Este modelo de infraestructura de clave pública se conoce como “Web de confianza”.
SPKI	Simple Public Key Infrastructure (Ellison et al., 1999) posee muchas similaridades con el modelo definido por PGP (cualquier poseedor de una

	clave puede emitir certificados). La gran diferencia se encuentra en que SPKI utiliza nombres de identidades relativos. Por ejemplo, si Pedro emite un certificado "Juan emitido por Pedro" podría ser o no ser la misma persona que "Juan emitido por María". Por este motivo, SPKI ha sido denominada como una PKI postmoderna. SPKI es todavía clasificada como una infraestructura experimental. Pese a ello, su influencia sobre otras propuestas es muy importante, como por ejemplo el marco de trabajo de seguridad definido por Microsoft para su plataforma .NET.
DNSSEC	DNS Security (Eastlake, 1999) es una PKI específica para asegurar los servicios de DNS. Extensiones futuras permitirán que DNSSEC sea utilizado para garantizar la seguridad de las aplicaciones una vez se haya probado que DNSSEC es en sí mismo seguro.

Tabla 1. Infraestructuras de Clave Pública más conocidas.

Para poder describir los mecanismos descritos por la especificación XKMS debemos poseer primero un conocimiento básico de las funcionalidades ofrecidas por una PKI.

El principal objetivo de XKMS es permitir a un programador utilizar una PKI sin la necesidad de que tenga que poseer un profundo conocimiento de lo que hace o de cómo lo hace. Por ello trataremos los conceptos básicos sobre PKI sin entrar en los detalles.

Una PKI gestiona credenciales. Una credencial es un conjunto de atributos asociados con un principal que se encuentran respaldados (por ejemplo firmados digitalmente) por un tercero (por ejemplo, una autoridad de confianza). En PKI, las credenciales indican el nombre del poseedor de la clave privada correspondiente a la clave pública.

Un nombre puede ser el nombre de una persona, de una compañía, un nombre de red (una dirección de correo electrónico), etc.

Antes de que una credencial pueda ser emitida en favor de un sujeto, la autoridad emisora deberá verificar que la parte solicitante es la poseedora legítima del nombre especificado y de que es la poseedora real de la clave privada asociada con la clave pública especificada.
Una vez se ha emitido la credencial, están podrán ser revocadas como por ejemplo cuando:

- La clave privada se ha visto comprometida (se ha perdido, fue revelada accidentalmente, etc.).
- Se averiguó que la información de la credencial es inválida.
- El poseedor de la clave ha roto los términos de uso impuestos por el emisor de la credencial. Por ejemplo, se emitió una credencial bajo la condición de que la clave privada sólo fuera utilizada para firmar cierto tipo de mensajes y, sin embargo, el poseedor la utilizó para firmar otros.

5.6.3 Relación de XKMS con PKI

Una PKI ofrece un mecanismo para vincular una identidad del mundo real con una identidad online. El concepto de identidad en el mundo real es muy complejo. Un mismo nombre puede ser utilizado por muchas personas o una persona puede emplear varios nombres. En el mundo de las empresas es todavía más complejo. Si no, pensemos en la cantidad de

equívocos surgidos por nombres muy parecidos de empresas absolutamente distintas.

XKMS no intenta eliminar la complejidad inherente a las plataformas PKI, tan sólo intenta transferir la complejidad de los clientes, donde esta complejidad resulta muy complicada de gestionar, a un servicio que se puede especializar en la gestión de la PKI.

Esta transferencia de complejidad ofrece grandes ventajas:

- Reduce la complejidad de los clientes. Uno de los objetivos iniciales de la especificación XKMS es permitir que los dispositivos móviles de baja capacidad de memoria y procesamiento pudieran acceder a toda la funcionalidad de una PKI.
- Facilita la codificación. En las PKI tradicionales los desarrolladores de las aplicaciones utilizaban algún tipo de conjunto de herramientas o capa del sistema operativo que les permitía dar soporte de funciones PKI. Esta aproximación normalmente es muy poco interoperable, ya que este conjunto de herramientas no cumple el estándar PKIX por completo (siempre existen ciertos elementos propietarios) de forma que la solución desarrollada queda vinculada con una solución prioritaria lo que degrada su nivel de conectividad e interacción con otros sistemas PKI.
- Problema de Configuración de los Clientes

  A medida que el uso de los PKI ha crecido, la complejidad de PKIX también ha aumentado considerablemente. Las implementaciones de estas propiedades más avanzadas se están llevando a cabo muy

lentamente. Uno de los principales motivos de esta lenta adopción es que hasta que esa característica no ha sido divulgada y es utilizada por una población significativa, agregarla a la solución PKI tan sólo incrementa su complejidad.

Pese a que existiera un acuerdo general entre los principales fabricantes de productos PKI para dotar a sus soluciones con una de estas funcionalidades, el periodo de despliegue de estas funcionalidades entre los clientes suele sufrir una dilatación muy considerable. Se estima, que este retraso es de alrededor de 10 años desde que se realiza el diseño inicial de la nueva funcionalidad PKI hasta que esta se ve divulgada ampliamente como para que las aplicaciones PKI clientes le den soporte.

XKMS elimina la complejidad de PKI del cliente y la transfiere a un servicio de confianza. Esto permite que las nuevas funcionalidades PKI sólo tengan que ser desplegadas en este servicio sin tener que realizar modificación alguna sobre los clientes.

XKMS es una solución ideal si se requiere dotar de algún tipo de soporte PKI a un dispositivo embebido como por ejemplo un enrutador de una Red Privada Virtual. Estos dispositivos tienen como requisito ser de “instalar y olvídate” ya que normalmente los clientes no quieren tener que actualizar regularmente o reemplazar

estos dispositivos con el objeto de mantener al día con los nuevos desarrollos realizados en el campo de las PKI.

5.6.4 Protocolo XKMS

El protocolo XKMS es un protocolo petición/respuesta basado en SOAP. La especificación XKMS define dos servicios que utilizan este protocolo: XML Key Registration Service (XKRSS) y XML Key Information Service (XML-KISS).

El servicio XML Key Registration Service (XKRSS) ofrece un conjunto de operaciones que permiten gestionar el ciclo de vida de las credenciales basadas en clave pública.

Por otro lado, también ofrece el servicio definido por la especificación XML Key Information Service (XML-KISS). Este servicio ofrece operaciones de petición que permiten obtener y validar credenciales de clave pública.

Antes de que veamos más detalladamente los mecanismos definidos por ambos servicios en relación con las credenciales de clave pública, realicemos una breve consideración sobre éstas últimas. El número de soluciones PKI es muy grande y se encuentra en continua evolución. Esta evolución hace que los tipos y formas de las credenciales sean muy diversos y que no sepamos con certeza cómo serán dentro de unos años. XKMS trata de dar soporte a cualquier tipo de PKI tanto actual como

futura, por ello, trata de ser independiente de la forma que tomen las credenciales para lo que define una super-credencial que permite adoptar cualquier tipo de credencial. Esta super-credencial es una asociación de vinculación de una clave. Esta credencial es una declaración o afirmación de que el poseedor de la clave privada, correspondiente a una clave pública especificada, se encuentra asociada con una o más identidades y direcciones de protocolos de Internet.

Esta definición es bastante abstracta lo cual permite acomodar cualquier modelo de credencial actual y futuro. Un ejemplo podría ser que Pedro posee una clave privada correspondiente a la clave pública E, de forma que se podría crear una asociación de vinculación de clave para reflejar este hecho. Con el objeto de comunicarse con usuarios de Internet, una asociación de vinculación de clave que asocia una clave pública con un nombre del mundo real resulta ser bastante menos útil que una que conecte esa clave pública con una dirección de correo electrónico, servicio chat online u otro forma de dirección de Internet.

XKMS define un elemento *KeyBinding* que permite definir una asociación de vinculación de clave que conecte una identidad de Internet con una clave pública. Este elemento permite adoptar cualquier tipo de credencial soportada por el elemento *ds:KeyInfo* definido en la especificación XML Digital Signature. Además, utiliza uno o más elementos *UseKeyWith* para asociar esa credencial con un uso determinado (un protocolo) de Internet.

XKMS define cuatro elementos que definen o bien una instancia real de una asociación de clave o bien una petición de obtención o creación de una asociación de clave:

– *KeyBinding*. Una instancia de una vinculación de clave emitida por una parte de confianza que ha sido validada por el usuario.
– *UnverifiedKeyBinding*. Una instancia de una vinculación de clave emitida por una parte de confianza que ha NO sido validada por el usuario.
– *QueryKeyBinding*. Una plantilla utilizada para solicitar una instancia de una vinculación de clave utilizando una petición de ejemplo.
– *PrototypeKeyBinding*. Una plantilla que especifica los parámetros solicitados a la hora de registrar una nueva vinculación de clave.

5.6.5 XKISS

Esta especificación define dos operaciones aplicables sobre las credenciales: localización y la validación. Ambas operaciones permite a un cliente obtener una credencial de un servicio XKMS. La diferencia entre ambas radica en que la primera operación retorna credenciales no verificadas (UnverifiedKeyBinding) mientras que la segunda operación retorna credenciales cuya validez ha sido verificada como por ejemplo que la credencial no ha caducado o no ha sido revocada.

5.6.6 XKRSS

Esta especificación define cuatro operativas relacionadas con el ciclo de vida de las credenciales: registrar, recuperar, revocar, y renovar.

El registro es el primer paso en el ciclo de vida de una credencial de vinculación de clave. En este paso el cliente especifica la información que desea vincular con la clave pública. El cliente puede solicitar al servicio de registro que genere el par clave pública-privada que va ser registrado (generación en el lado del servidor) o puede causar que el par sea generado por él mismo (generación en el lado de cliente).

La aproximación de generar el par en el lado de cliente evita que tenga que ser desvelada la clave privada al servicio de registro. Esta aproximación, siempre que sea posible, es preferible frente a la generación del par en el lado del servidor. Normalmente esta opción se utiliza cuando el dispositivo diente es limitado y no tiene suficiente capacidad para generar o almacenar las claves.

Durante el registro de una nueva vinculación de clave el cliente realiza en primer lugar una petición en la que incluye un prototipo de la vinculación de clave que se desea sea emitida, un elemento de autenticación en el caso de que el par haya sido generado en el lado del cliente, y una prueba de posesión del elemento. Si la operación de registro se ejecuta correctamente, la respuesta contiene una vinculación de clave resultado

y, en el caso de generar el par en el lado del servidor, la clave privada cifrada.

La operación de recuperación permite la obtención de una copia de la clave privada en aquellos casos en los que el posesor la haya perdido por algún motivo (por ejemplo cuando va embebida en una tarjeta inteligente y éste se rompe). Esta operación implica que el servicio dispone de una copia de la clave privada lo cual, desde el punto de vista de la seguridad, impide el no repudio ya que más de una entidad posee la clave privada. Normalmente este servicio de recuperación se emplea en servicios de ámbito gubernamental.

La operación de renovación permite, como la propia palabra expresa, la renovación de una vinculación de clave. Esta renovación permite que la credencial en cuestión sea actualizada periódicamente y además permite revisar y comprobar la validez de la confianza puesta en la asociación otorgada entre la clave y la información vinculada. Esta petición de renovación contiene la misma información que la contenida en una petición de registro.

La operación de revocación permite revocar la validez de un vínculo de clave previamente emitido. Revocar una vinculación de clave significa que ésta deja de poder considerarse de confianza. Un vínculo de clave puede ser revocada por multitud de motivos: la clave privada ha sido revelada o comprometida de alguna forma, se puede descubrir que hubo un error tras su emisión, etc.

6. ESPECIFICACIONES DE SEGURIDAD DE OASIS

6. Especificaciones de Seguridad de OASIS

6.1 WS-Security

Organización	OASIS
Estado	Liberada
Versión	1.0 (19/4/2004)

6.1.1 Introducción

La versión 1.0 (OASIS, 2004) de esta especificación emitida por el comité de seguridad en servicios Web del consorcio OASIS se compone de tres documentos más dos esquemas XML.

WS-Security describe mejoras al marco de trabajo de mensajería SOAP con el fin de proporcionar la propiedad de calidad de protección (QoP) mediante dos mecanismos esenciales:

- La integridad y confidencialidad de los mensajes.
- Autenticación de un mensaje individual.

Estos mecanismos se pueden utilizar para acomodar una amplia variedad de modelos de seguridad y tecnologías de cifrado. WS-Security también proporciona un mecanismo de propósito general para asociar tokens de seguridad con mensajes. Un token de seguridad agrupa un conjunto de declaraciones de seguridad. No es necesario ningún tipo específico de

token de seguridad para utilizar WS-Security ya que está diseñado para ser extensible, soportando múltiples formatos de tokens de seguridad.

Adicionalmente, WS-Security describe cómo codificar tokens de seguridad binarios. Específicamente describe cómo codificar certificados X.509 y tickets de Kerberos así como la inclusión de claves cifradas opacas (p.e.: el resultado de aplicar una función MD5 sobre una password). También incluye mecanismos de extensibilidad que pueden ser utilizados para describir las características de las credenciales que se incluyen en un mensaje.

Gracias al uso del modelo de extensibilidad de SOAP, las especificaciones que están basadas en dicha tecnología están diseñadas para que, al complementarse unas con otras, se puedan crear entornos de mensajería completos. Por sí misma, la especificación WS-Security no asegura la seguridad ni es capaz de proporcionar una solución de seguridad completa. WS-Security es un **bloque de construcción** que es utilizado conjuntamente con otros protocolos de servicios Web o específicos de aplicación para acomodar una amplia variedad de modelos de seguridad y tecnologías de cifrado. Implementar esta especificación no significa que una aplicación no pueda ser atacada o que la seguridad no pueda verse comprometida alguna vez.

WS-Security proporciona un estándar de extensiones del entorno de mensajería SOAP que puede ser utilizado cuando se construyen servicios Web seguros que necesitan confidencialidad e integridad. Nos referimos

a este conjunto de extensiones como “Lenguaje de Seguridad de Servicios Web” o “WS-Security”.

WS-Security es flexible y su diseño constituye la base para la creación de modelos de seguridad más complejos incluyendo PKI, Kerberos y SSL. En particular, WS-Security proporciona soporte para múltiples tokens de seguridad, múltiples dominios de confianza, múltiples formatos de firma y múltiples tecnologías de cifrado. Esta especificación proporciona 3 elementos principales:

1. Propagación de tokens de seguridad.
2. Confidencialidad de los mensajes.
3. Integridad de los mensajes.

Estos mecanismos por sí solos no proporcionan una solución completa de seguridad. Sin embargo, WS-Security es un bloque de construcción que utilizado con otras extensiones de servicios Web o con protocolos propietarios de más alto nivel de aplicación es capaz de acomodar una amplia variedad de modelos de seguridad y tecnologías de cifrado.

Estos mecanismos pueden ser utilizados independientemente (p.e: para pasar un token de seguridad) o de una manera altamente integrada (p.e: firmando y cifrando un mensaje y proporcionando una jerarquía de elementos de seguridad asociada con las claves utilizadas para el firmado y el cifrado).

La meta de WS-Security es permitir que las aplicaciones construyan intercambios seguros de mensajes SOAP. Esta especificación está destinada a proporcionar un conjunto flexible de mecanismos que puedan ser utilizados para construir una amplia gama de protocolos de seguridad; en otras palabras, esta especificación intencionadamente no describe explícitamente protocolos de seguridad sino que define las primitivas para poder hacerlo.

Como con cualquier protocolo de seguridad, WS-Security por sí solo no garantiza la seguridad y se debe aplicar un gran esfuerzo para garantizar que los protocolos de seguridad construidos con WS-Security no resulten vulnerables cuando se enfrentan ante un amplio abanico de ataques.

Como ya se ha dicho, el Lenguaje de Seguridad de Servicios Web (WS-Security) soporta una amplia variedad de modelos de seguridad. La siguiente lista identifica los requisitos clave que han conducido el desarrollo de la especificación:

- Múltiples tokens de seguridad para autenticación y autorización.
- Múltiples dominios de seguridad.
- Múltiples tecnologías de seguridad.
- Seguridad a nivel de mensaje de extremo a extremo de la comunicación y no tan sólo seguridad punto a punto a nivel de transporte.

6.1.2 Ejemplo

El siguiente ejemplo ilustra un mensaje con un token de seguridad de nombre de usuario:

```
<?xml version="1.0" encoding="utf-8" ?>
<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope"
      xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
   <S:Header>
      <m:path xmlns:m="http://schemas.xmlsoap.org/rp/">
            <m:action>http://fabrikam123.com/getQuote</m:action>
            <m:to>http://fabrikam123.com/stocks</m:to>
            <m:id>uuid:84b9f5d0-33fb-4a81-b02b-5b760641c1d6</m:id>
        </m:path>
      <wsse:Security
            xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext">
         <wsse:UsernameToken Id="MyID">
            <wsse:Username>Zoe</wsse:Username>
         </wsse:UsernameToken>
         <ds:Signature>
            <ds:SignedInfo>
               <ds:CanonicalizationMethod
                  Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
                  />
               <ds:SignatureMethod
                  Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-
                  sha1" />
               <ds:Reference URI="#MsgBody">
                  <ds:DigestMethod
                     Algorithm="http://www.w3.org/2000/09/xmldsig#sha1
                     " />
                  <ds:DigestValue>LyLsF0Pi4wPU...</ds:DigestValue>
               </ds:Reference>
            </ds:SignedInfo>
            <ds:SignatureValue>DJbchm5gK...</ds:SignatureValue>
            <ds:KeyInfo>
               <wsse:SecurityTokenReference>
                  <wsse:Reference URI="#MyID" />
               </wsse:SecurityTokenReference>
            </ds:KeyInfo>
         </ds:Signature>
      </wsse:Security>
   </S:Header>
   <S:Body Id="MsgBody">
      <tru:StockSymbol
         xmlns:tru="http://fabrikam123.com/payloads">QQQ</tru:StockSy
         mbol>
   </S:Body>
</S:Envelope>
```

Podemos ver cómo en el bloque de cabecera SOAP aparece el elemento *Security* que se define en la especificación. Esta cabecera contiene información de seguridad para cierto receptor. Dentro del elemento *Security* se define un token de seguridad, en este caso de tipo *UsernameToken* que define el nombre del cliente emisor del mensaje. Destacar que en este ejemplo se asume que el servicio receptor del mensaje conoce el secreto compartido y que está asociado con el nombre del usuario dado.

Además del token de seguridad, dentro de esta cabecera WS-Security se define una firma acorde con la especificación W3C XML Digital Signature. Esta firma asegura la integridad de los elementos firmados. En este ejemplo la firma está basada en una clave generada a partir de la contraseña del usuario; normalmente se utilizan mecanismos de firma más robustos (como veremos más adelante en el ejemplo extendido).

Dentro del elemento *KeyInfo* se encuentra un elemento *SecurityTokenReference* que referencia al token de seguridad *UsernameToken* indicado anteriormente y que está asociado con la firma. Por ultimo, hacer notar cómo la última parte contiene el cuerpo del mensaje SOAP con el contenido de negocio real.

6.1.3 Calidad de protección (Quality of Protection)

Con el fin de asegurar un mensaje SOAP, se deben tener en cuenta dos posibles amenazas:

1. El mensaje podría ser modificado o leído por un atacante.
2. Un atacante podría enviar mensajes a un servicio de forma que, aunque está bien construido, carece de las declaraciones de seguridad apropiadas para garantizar el proceso.

Para entender estas amenazas WS-Security define un modelo de seguridad de mensajes.

6.1.4 Modelo de Seguridad de Mensajes

En este documento se especifica un modelo de seguridad de mensajes abstracto basado en tokens de seguridad combinados con firmas digitales que prueban la posesión del token de seguridad (p.e.: una clave privada o un secreto compartido).

Los tokens de seguridad contienen afirmaciones y las firmas proporcionan un mecanismo para probar el conocimiento de la clave secreta por parte del emisor. También, la firma puede ser utilizada para "vincular" o "asociar" la firma con las afirmaciones del token de seguridad. Obviamente, tal vinculación se limita a aquellos elementos cubiertos por la firma. Es más, es necesario destacar que la especificación

WS-Security no especifica un método particular (ya hemos dicho que define un modelo abstracto de seguridad) para el proceso de autenticación, sino que simplemente ofrece la posibilidad de asociar tokens de seguridad a los mensajes SOAP.

Una afirmación puede ser **respaldada** (la especificación denomina a este tipo de afirmación *'endorsed'*) o no por una autoridad. Un conjunto de afirmaciones respaldadas se representan normalmente mediante un token de seguridad que es digitalmente firmado o cifrado por la autoridad que ofrece el respaldo. Un certificado X.509, que confirma la vinculación existente entre la identidad de alguien y una clave pública, es un ejemplo de un token de seguridad respaldado. Una afirmación respaldada también puede ser representada como una referencia a una autoridad de forma que el receptor puede obtener la afirmación desde la autoridad referenciada.

Una afirmación no respaldada puede ser confiable si existe una relación de confianza entre el emisor y el receptor. Por ejemplo, la afirmación no respaldada que dice que el emisor es Bob puede ser suficiente para que cierto receptor piense que el emisor es realmente Bob, siempre y cuando el emisor y el receptor utilicen una conexión confiada y exista una relación de confianza acordada "offline" entre ellos. Un tipo especial de afirmación no respaldada es la afirmación de 'prueba de posesión'. Tal afirmación prueba que el emisor posee una pieza de conocimiento particular que es verificable por los actores apropiados. Por ejemplo, un nombre de usuario y contraseña es un token de seguridad con este tipo de afirmación. Sólo ciertos servicios serán capaces de verificar que la

contraseña está realmente asociada con el nombre de usuario y la contraseña, además, no se encuentra respaldada por nadie y, sin embargo, sirve como afirmación de prueba de posesión de que la entidad cliente tiene el nombre de usuario indicado. Como vemos a raíz del ejemplo anterior, algunas veces, una afirmación de prueba de posesión se combina con otros tokens de seguridad para probar las afirmaciones incluidas por el emisor en ese token.

6.1.5 Protección de Mensaje

Proteger el contenido del mensaje de posibles intercepciones (confidencialidad) o de modificaciones ilegales (integridad) son las preocupaciones básicas de la seguridad. La especificación WS-Security proporciona un medio para proteger un mensaje cifrando y/o firmando el cuerpo, una cabecera, un anexo o cualquier combinación de ellos (o partes de ellos).

La integridad de los mensajes se consigue aplicando los mecanismos definidos por la especificación W3C XML Digital Signature conjuntamente con el empleo de tokens de seguridad, combinación que permite asegurar que los mensajes son transmitidos sin modificaciones. Los mecanismos de integridad están diseñados para soportar múltiples firmas, potencialmente de múltiples actores, y ser extensibles para soportar formatos de firmas adicionales.

La confidencialidad de los mensajes SOAP se consigue mediante la aplicación de los mecanismos definidos por la especificación W3C XML

Encryption en combinación con los tokens de seguridad. Los mecanismos de cifrado están diseñados para soportar procesos de cifrado adicionales que surjan en un futuro así como operaciones realizadas por múltiples actores.

Cuando un recipiente de un mensaje WS-Security no puede verificar la firma, el token de seguridad recibido no contiene las afirmaciones suficientes o algunas de ellas no contiene los valores esperados, la especificación recomienda que el mensaje sea rechazado.

6.1.6 Referencias ID

Esta especificación define el atributo *wsu:Id* de forma que los receptores puedan hacer referencia a elementos arbitrarios de un mensaje como por ejemplo una firma digital. Sin embargo, puesto que algunos esquemas XML clave utilizados por esta especificación, como aquellos definidos por la especificación W3C XML Digital Signature o W3C XML Encryption, no permiten extensibilidad de los atributos, esta especificación también permite el uso de sus atributos ID locales además del atributo *wsu:Id*. En consecuencia, cuando se intenta localizar un elemento referenciado en una firma (recordar el elemento *ds:Reference*), se pueden considerar los siguientes atributos:

- Atributos ID locales contenidos en los elementos definidos por la especificación W3C XML Digital Signature.
- Atributos ID locales de los elementos definidos por la especificación W3C XML Encryption.

- Los atributos globales referenciados utilizando el atributo *wsu:Id*, que se describirá a continuación.

Además, cuando se firma una parte de un sobre SOAP, como por ejemplo el cuerpo, se recomienda que se utilice una referencia de identificación en vez de una transformación más general, en concreto una transformación XPath. El principal y único motivo es simplificar el procesamiento.

Como ya se ha dicho, existen muchas situaciones en las que se necesita referenciar ciertos elementos contenidos dentro de los mensajes SOAP. Por ejemplo, cuando se firma un mensaje SOAP, los elementos seleccionados se incluyen en el ámbito de la firma. En la parte 2 de la especificación W3C XML Schema se proporcionan varios tipos de datos que podrían utilizarse para identificar y referenciar estos elementos. El problema de utilizar estos tipos de datos es que se obliga a los consumidores de los mensajes SOAP a disponer, o a ser capaces de obtener, los esquemas donde se encuentran definidos la identidad o los mecanismos de referencia. En algunos casos, como por ejemplo en los actores que actúan como intermediarios, esto puede ser problemático y muy poco deseable.

En consecuencia, los desarrolladores de esta especificación vieron necesario definir un mecanismo para identificar y referenciar elementos, basado en el núcleo de la especificación SOAP, el cual no dependiera del conocimiento completo y explícito del contexto en el que el elemento es

utilizado. Esta funcionalidad puede ser integrada en los procesadores SOAP de forma que los elementos puedan ser identificados y referenciados sin necesidad de descubrir ni procesar ningún esquema dinámicamente.

Esta especificación define un atributo global completamente cualificado que sirve para identificar un elemento y que puede ser aplicado sobre cualquier elemento que, o bien permita atributos, o bien permita específicamente un atributo en particular.

Para simplificar el procesamiento de los intermediarios y de los receptores finales, se define un atributo común para identificar cualquier elemento. Este atributo utiliza el tipo ID definido por la especificación W3C XML Schema.

La sintaxis de este atributo es:

```
<anyElement wsu:Id="...">...</anyElement>
```

El atributo '*@wsu:Id* ', es del tipo xsd:ID, y proporciona un atributo para especificar el identificador local de un elemento. Por su propia naturaleza, dos atributos *wsu:Id* dentro de un documento XML no podrán tener el mismo valor.

La especificación en sí no especifica cómo se debe utilizar este atributo, y se espera que sean otras especificaciones las que añadan semánticas

adicionales (o ciertas restricciones) para su uso. El siguiente ejemplo ilustra el uso de este atributo para identificar un elemento:

```
<x:myElement wsu:Id="ID1" xmlns:x="..." xmlns:wsu="..." />
```

Aquellos procesadores conformes con la especificación que soporten la especificación W3C XML Schema deben tratar este atributo como si hubiera sido definido mediante una declaración global.

Por otro lado, la especificación recomienda que los procesadores conformes con esta especificación que no soporten el descubrimiento y el procesamiento dinámico de esquemas XML o DTDs integren la definición de este atributo en sus analizadores.

6.1.7 Elemento Security

El bloque de cabecera SOAP *Security* (omitimos el prefijo 'wsse:' comúnmente utilizado) proporciona un mecanismo para asociar información de seguridad destinada a un receptor específico (*actor o rol SOAP*). Este actor podría ser el último receptor del mensaje o un simple intermediario. En consecuencia, este bloque de cabecera podría estar presente varias veces en un mismo mensaje SOAP. Un intermediario en el camino del mensaje SOAP podría agregar uno o más sub-elementos a un bloque de cabecera *Security* existente en el caso de que estén destinados al mismo nodo SOAP; o podría agregar una o más cabeceras nuevas para destinatarios adicionales.

Como ya se ha mencionado, un mensaje podría tener múltiples bloques de cabecera *Security* si están destinados a distintos receptores. Sin embargo, solamente uno de estos bloques de cabecera puede omitir el atributo *SOAP:role* (*SOAP:actor* en la versión SOAP 1.1) y no puede haber dos de estos bloques de cabecera con el mismo valor en el atributo *SOAP:role*. La información de seguridad destinada a los distintos receptores debe aparecer en bloques de cabecera *Security* separados. Un bloque de cabecera de este tipo sin un *SOAP:role* definido puede ser consumido por cualquiera, pero no debe ser eliminado en ningún nodo SOAP previo al último nodo SOAP destinatario tal y como se especifica en WS-Routing.

A medida que se añaden elementos al bloque de cabecera *Security*, deberían ser antepuestos a los elementos existentes. De esta forma, el bloque de cabecera *Security* representa los pasos de firmado y cifrado que el emisor realizó para crear el mensaje. Esta regla de anteposición asegura que la aplicación receptora pueda procesar los sub-elementos en el orden en el que aparecen en el bloque de cabecera *Security*, y de esta forma no encontrarse más adelante con dependencias entre los sub-elementos. La especificación no obliga a seguir ningún orden específico a la hora de procesar los sub-elementos pudiendo utilizar la aplicación receptora cualquier política que crea necesaria.

Cuando un sub-elemento hace referencia a una clave transportada en otro sub-elemento (por ejemplo un sub-elemento firma que hace referencia a un sub-elemento token de seguridad binario que contiene un certificado

X.509 utilizada para verificar la firma), el token de seguridad portador de la clave debería estar antepuesto al sub-elemento que utiliza la clave y que está siendo añadido, de forma que el material de clave aparezca con anterioridad al sub-elemento que utiliza la clave. El siguiente documento XML ilustra la sintaxis de la cabecera:

```
<S:Envelope>
  <S:Header>
    ...
      <Security S:role="..." S:mustUnderstand="...">...</Security>
    ...
  </S:Header>
  ...
</S:Envelope>
```

Todas las implementaciones que cumplan esta especificación deben ser capaces de procesar un elemento *Security.* Además, todas las implementaciones conformes deben declarar explícitamente qué perfiles soportan y deben ser capaces de procesar elementos *Security* incluyendo cualquier sub-elemento que pueda ser definidor por ese perfil.

Cuando una cabecera *Security* incluye un atributo *mustUnderstand=”true” (definido por la especificación SOAP 1.x)*:

- El receptor debe generar un fallo SOAP si no implementa esta especificación de acuerdo con el espacio de nombres. Implementación significa que el receptor no es capaz de

interpretar el esquema ni de seguir las reglas de procesamiento requerido establecidos por esta especificación.

- El receptor debe generar un fallo si es incapaz de interpretar o procesar el token de seguridad contenido en el bloque de cabecera *Security* de acuerdo a algunos de los perfiles definidos por esta especificación.
- Los receptores pueden ignorar elementos o extensiones dentro del elemento *Security*, en función a su política local.

6.1.8 Token de Seguridad

Como ya se ha definido anteriormente un token de seguridad es una colección de afirmaciones hechas por una entidad. Para transmitir tokens de seguridad (es decir afirmaciones de seguridad realizadas por un interlocutor) la especificación propone el elemento *Security* de forma que cualquier token de seguridad que se desee transmitir sea un elemento hijo de éste. Por su diseño, el elemento *Security* es extensible para soportar muchos tipos de información relativos con la seguridad.

Esta especificación, además de definir ciertos tokens de seguridad, define las reglas de procesamiento para utilizar los elementos definidos en la especificación W3C XML Digital Signature y XML Encryption. Si se utiliza cifrado o firma digital en combinación con tokens de seguridad, éstos últimos deberán ser utilizados de forma que se respeten las reglas de procesamiento definidas por esta especificación.

6.1.9 Elemento UsernameToken

Anteriormente en este apartado ya hemos introducido el elemento *UsernameToken* como una forma de probar un nombre de usuario y una información de contraseña adicional. El documento *UsernameToken Profile 1.0* que forma parte de la versión 1.0 del estándar WS-Security define este tipo de token en detalle.

El elemento *UsernameToken* describe el método por el cual un consumidor de un servicio Web puede proporcionar un elemento *UsernameToken* como medio para identificar al solicitante mediante un nombre de usuario y, opcionalmente, una contraseña (o una clave secreta) para autenticar esa identidad de cara al proveedor del servicio.

La sintaxis es muy sencilla:

```
<UsernameToken wsu:Id="...">
  <Username>...</Username>
  <Password>...</Password>
</UsernameToken>
```

El elemento *UsernameToken* se utiliza para representar una declaración de una identidad. Aquí vemos un ejemplo del uso del atributo global *wsu:Id*. Se puede utilizar este atributo para poder identificar local y unívocamente este token de seguridad. El sub-elemento *Username* contiene una cadena con la identidad declarada. Se puede agregar cualquier atributo, definido en cualquier otro esquema externo al elemento *Username*, siendo éste uno de los dos puntos posibles de extensibilidad de este elemento. El otro punto de extensibilidad es aquel

que permite incorporar cualquier tipo de elemento XML definido en un esquema propio como elemento hijo de *UsernameToken*.

6.1.10 Codificación de Tokens de Seguridad Binarios

Cualquier token de seguridad basado en XML puede ser especificado en la cabecera *Security*. Sin embargo, cuando el formato del token es binario u otro que no sea XML (certificados X.509 o tickets de Kerberos) se requiere un formato de codificación especial para que pueda ser incluido.

Un token de seguridad binario tiene dos atributos que son utilizados para poder interpretarlo. El atributo *ValueType* indica de qué tipo es el token de seguridad, por ejemplo, un ticket de Kerberos. El atributo *EncodingType* indica la forma en la que el token de seguridad es codificado, como por ejemplo *Base64Binary* para codificación en base 64.

Por lo tanto, el elemento *BinarySecurityToken* define un token de seguridad que es codificado en binario. El uso de los dos atributos mencionados define la codificación utilizada. El siguiente fragmento muestra su sintaxis:

```
<BinarySecurityToken wsu:Id="..." EncodingType="..." ValueType="..." />
```

El atributo *EncodingType*, puede tener como valor una URI que indica el formato de codificación de los datos binarios (p.e.: codificación en base 64). La única URI, relativa a la definida por esta especificación, definida

es '#Base64Binary' que indica que el token de seguridad está codificado en base 64 según la especificación XML Schema. El atributo *ValueType* se utiliza para indicar el espacio de valores, mediante una URI, de los datos codificados en binario. Esta especificación no define ninguna URI concreta relegando dicha responsabilidad a otras especificaciones futuras.

6.1.11 Elemento SecurityTokenReference

Un token de seguridad transporta un conjunto de afirmaciones. Algunas veces, estas afirmaciones residen en otros lugares y necesitan ser obtenidas por la aplicación receptora. El elemento *SecurityTokenReference* proporciona un mecanismo extensible para referenciar tokens de seguridad de forma que la aplicación receptora pueda obtener el token a partir de su referencia. En el caso de que se utilizara este elemento fuera de un bloque de cabecera Security, el elemento que lo contuviera sería responsable de definir su conducta, cosa que queda fuera del alcance de la especificación. La sintaxis es muy sencilla como se puede ver a continuación:

```
<SecurityTokenReference wsu:Id="..." wsse:Usage="..." />
</SecurityTokenReference>
```

El significado del atributo *wsu:Id* es el de proporcionar un identificador único a la referencia y, en ningún caso, debe interpretarse como el valor mismo de la referencia. Para ello se utiliza un fragmento de URI incluido en un elemento *Reference* que debe estar incluido dentro del *SecurityTokenReference*. Existe un atributo opcional, *wsse:Usage*, que

contiene una URI que representa el tipo de uso (semántica) del elemento *SecurityTokenReference*. En su versión 1.0, la especificación no define ninguna URI en concreto. Como punto de extensibilidad, la especificación permite la inclusión de cualquier tipo de elemento XML como elemento hijo de *SecurityTokenReference*. En el caso en que una parte receptora no supiera interpretar dicho elemento deberá generar un fallo. De la misma manera, el elemento *SecurityTokenReference* soporta cualquier tipo de atributo definido por el usuario. En el caso de que un procesador del elemento no entendiera dicho atributo se deberá generar un error.

Este elemento podría ser también un hijo directo del elemento *ds:KeyInfo* con el fin de referenciar la información que indica como recuperar la clave de un token de seguridad ubicado en cualquier otro lugar. En particular, la especificación recomienda que, cuando se utilice XML Signature y XML Encryption, se coloque un elemento *SecurityTokenReference* dentro de un elemento *ds:KeyInfo* para así referenciar el token de seguridad utilizado para firmar o cifrar. Para ciertos tipos de tokens de seguridad, tales como certificados X.509, ambos, el elemento *ds:KeyInfo* y el elemento *BinarySecurityToken* pueden ser utilizados para transportar la información de la clave. El elemento *ds:KeyInfo* permite distintos tipos de claves para futuras extensiones. Sin embargo, la especificación WS-Security recomienda el uso del elemento *BinarySecurityToken* como mecanismo para transportar el material de las claves si el tipo de clave está entre los definidos por la especificación (valores del atributo *ValueType*).

La especificación posee ciertos retos que deberán ser afrontados por aquellos que quieran implementar esta especificación. El procesamiento de los identificadores y de las referencias, requiere que el recipiente entienda el esquema. Esto puede ser un coste computacional muy caro y en el caso general parece imposible que existe una manera de conocer la "ubicación del esquema" para una URI que representa un espacio de nombres específico. También, y como resulta lógico, el objetivo principal de una referencia es identificar unívocamente el token deseado. Las referencias ID son, por definición, únicas por XML. Sin embargo, otros mecanismos como por ejemplo el "nombre de un principal" no requieren ser unívocos y, por lo tanto, este tipo de referencias podrían no ser únicas. La siguiente lista proporciona una lista, ordenada de mayor a menor preferencia, de los mecanismos para referenciar mencionados por la especificación:

- **Referencias directas.** Si el token está incluido se utiliza un fragmento URI y si es externo una URI completa.
- **identificador de clave.** Permite que el token sea referenciado mediante un valor opaco que representa el token (definido por un tipo de token o un perfil).
- **Nombres de Clave.** Esto permite que los tokens sean referenciados mediante cadenas de texto que coinciden con alguna cadena de texto incluida en alguna afirmación de identidad incluida en el token.
- **Referencias incrustadas.** Esto permite que los tokens sean incrustados (por lo contrario a disponer de un puntero a un token que reside en otro lugar).

Referencias directas

El elemento *Reference* proporciona un mecanismo de extensibilidad que permite referenciar directamente tokens de seguridad mediante una URI. En este caso el elemento *Reference* incluiría un atributo URI y *ValueType*:

```
<SecurityTokenReference wsu:Id="...">
        <Reference URI="..." ValueType="..." />
</SecurityTokenReference>
```

La URI, que es un atributo opcional, especifica una URI abstracta donde se encuentra ubicado el token de seguridad. Si se especificara un fragmento, entonces se estará indicando el ID local del token que está siendo referenciado. Por su parte, el atributo *ValueType* define una URI que representa el tipo de token que se está referenciando. Un ejemplo de este tipo de referencia tokens sería el siguiente fragmento:

```
<wsse:SecurityTokenReference xmlns:wsse="...">
      <wsse:Reference URI="http://www.example.com/tokens/Zoe" />
</wsse:SecurityTokenReference>
```

Referencias mediante identificadores de claves

Si no se utiliza una referencia directa, la especificación recomienda que se utilice un identificador de clave para referenciar un token de seguridad en vez de utilizar el elemento *ds:KeyName* definido por la especificación XML Digital Signature. La especificación define el elemento *KeyIdentifier* que puede ser utilizado para identificar unívocamente un token de seguridad. El tipo del valor y el algoritmo de generación exactos

variarán según el tipo de token de seguridad (y, en algunas ocasiones, por los datos incluidos dentro del token). En consecuencia, los valores y algoritmos están descritos en los perfiles dedicados a los tipos de tokens específicos en vez de estar definidos por la especificación. El elemento *KeyIdentifier* se debería colocar como elemento hijo del elemento *SecurityTokenReference* con el fin de que el primero referencie un token de seguridad mediante un identificador. El modelo de procesamiento asume que el identificador de clave para un token de seguridad es constante e inmutable. En consecuencia, el procesamiento de un identificador de clave consiste simplemente en buscar un token de seguridad cuyo identificador de clave coincida con una constante especificada dada. El siguiente fragmento muestra la sintaxis de este elemento:

```
<wsse:SecurityTokenReference xmlns:wsse="...">
   <wsse:KeyIdentifier wsu:Id="..." ValueType="..." EncodingType="...">...
   </wsse:KeyIdentifier>
</wsse:SecurityTokenReference>
```

El elemento *KeyIdentifier* contendrá un identificador a un token de seguridad codificado en binario. El atributo opcional *ValueType* representa el tipo de identificador de clave que está siendo utilizado. Lo perfiles de token que sean creados deberán indicar los tipos de *KeyIdentifiers* que podrían ser utilizados para referenciar tokens de ese tipo. También deberán especificar la semántica crítica del identificador, como por ejemplo si el *KeyIdentifier* es o no único para la clave o el token. El atributo opcional *EncodingType* contiene una URI y se utilizará

para identificar el tipo de codificación que se está empleando para representar el identificador de la clave. La especificación define sólo el tipo de codificación binaria en base 64.

Nombres de Claves

La especificación recomienda que se utilice el elemento *KeyIdentifier*. Sin embargo, si se deben emplear nombres de claves, recomienda que los elementos *ds:KeyName* sean conformes a los nombres de atributos definidos en la sección 2.3 del RFC 2253 con el fin de obtener la mayor interoperabilidad posible.

Referencias incrustadas

En algunos escenarios la referencia podría ser a un token incrustado (en oposición a una referencia a un token que reside en otro lugar). Para hacer esto, se utilizará el elemento *Embedded* como elemento hijo del elemento *SecurityTokenReference*. La sintaxis de este elemento es la siguiente:

El elemento *Embedded* se utilizará para incrustar directamente el token en la referencia. En el siguiente ejemplo se muestra como se incrustaría una afirmación SAML dentro de una referencia:

```
<wsse:SecurityTokenReference xmlns:wsse="...">
    <wsse:Embedded wsu:Id="...">...</wsse:Embedded>
</wsse:SecurityTokenReference>
```

ds:KeyInfo

El elemento *ds:KeyInfo* (definido en la especificación XML Digital Signature) puede ser utilizado para transportar la información de la clave. Sin embargo, en esta especificación, el mecanismo que se recomienda es el uso del token *BinarySecurityToken* para transportar el material de clave en el caso de que la clave contenga datos binarios. El siguiente ejemplo, ilustra el uso de este elemento:

```
<ds:KeyInfo Id="..." xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:KeyName>CN=Pedro Gomez, C=ES</ds:KeyName>
</ds:KeyInfo>
```

6.1.12 Firmas Digitales

Un productor de mensajes podría querer otorgar la capacidad a los receptores de sus mensajes de verificar que los mensajes enviados por él no han sido modificados durante su tránsito y que las declaraciones contenidas dentro de un token de seguridad realmente le pertenecen. Por ejemplo, cierto emisor E podría enviar un mensaje M a cierto receptor R. El mensaje M contiene una firma digital realizada por E sobre el contenido de forma que R cuando lo reciba podrá verificar la integridad del mensaje así como que fue E quién realmente lo envió.

La especificación WS-Security permite ligar múltiples firmas digitales con un mensaje, cada una referenciando partes diferentes, incluso solapadas, de un mismo mensaje. Esto es importante para numerosas aplicaciones distribuidas donde los mensajes fluyen por múltiples etapas de procesamiento. Por ejemplo, un emisor podría enviar un pedido que

contiene una cabecera con el identificador del pedido pedidoID. El emisor firma la cabecera pedidoID y el cuerpo de la petición (que son los contenidos del pedido). Cuando ésta es recibida por el sub-sistema de procesamiento de pedidos, podría insertar un identificador embarcadoID dentro de la cabecera, y posiblemente también en el cuerpo. Entonces, cuando el pedido es procesado y enviado por el departamento de envíos, se agrega una cabecera más, embarcadoInfo. El departamento de envíos firmaría, como mínimo, las cabecera embarcadoInfo y embarcadoID y, posiblemente, el cuerpo, y reenviaría el mensaje al departamento de facturación para su proceso. El departamento de facturación puede verificar las firmas y determinar una cadena de confianza para el pedido así como conocer quién hizo qué.

Debido a la mutabilidad de las cabeceras SOAP, la especificación WS-Security recomienda no utilizar la transformación 'Enveloped Transform Signature' definida en la especificación XML Digital Signature a la hora de generar la firma. Las firmas digitales 'envueltas' son aquellas en las que los datos firmados contienen la firma digital:

```
<datos>
        ...
    <ds:Signature>...</ds:Signature>
        ...
</datos>
```

Por el mismo motivo, los productores de mensaje tampoco deben utilizar firmas digitales envolventes, es decir, aquellas que contienen internamente los datos que son firmados:

```
<ds:Signature>
        ...
        <datos>
        ...
       </datos>
</ds:Signature>
```

Los datos que son firmados deben estar incluidos explícitamente en el propio mensaje.

Algoritmos

WS-Security se construye sobre *XML Signature* y, por tanto, debe soportar los mismos requisitos que ésta en cuanto a los algoritmos se refiere.

El algoritmo *Exclusive XML Canonical* cubre las trampas que tiene la normalización general que proviene de las fugas de los espacios de nombres con firmas pre-existentes. Debido a los problemas de seguridad con los espacios de nombre, la especificación WS-Security recomienda el uso del algoritmo *Exclusive XML Canonicalization* u otro algoritmo de normalización que proporcione la misma, o superior, protección.

Finalmente, el emisor que desee firmar un mensaje antes de cifrarlo, deberá comprobar que la estructura del bloque de cabecera Security contenga en primer lugar el elemento cifrado y en segundo lugar la firma digital:

Esta disposición no hace más que seguir la regla de anteposición descrita durante la definición del elemento Security realizada anteriormente. De la misma manera, si el productor del mensaje desea cifrar los datos en primer lugar y luego firmarlos, deberá seguir la misma regla quedando como primer elemento hijo de la cabecera Security el elemento correspondiente a la firma digital y como segundo hijo el elemento que contenga los datos cifrados.

La especificación W3C XML Digital Signature define dos métodos de normalización de la información que es firmada: XML Canonicalization y XML Exclusive Canonicalization. El primer método se conoce también como el método de normalización inclusivo mientras que el segundo se conoce como el método de normalización exclusivo. En relación con el formato de la información que es firmada digitalmente existen dos problemas a resolver. El primero guarda relación con la equivalencia lógica entre documentos XML estructuralmente distintos. Dos documentos XML que no son absolutamente idénticos byte a byte podrían ser lógicamente equivalentes y es deseable que ambos produzcan la misma firma digital octeto a octeto. Para ello se debe aplicar un mecanismo de normalización sobre los datos a firmar de forma que, pese a ser distintos en el origen, se conviertan en idénticos en su forma normalizada.

El segundo problema es que las firmas digitales podrían desear firmar datos como xx:foo. El significado de xx:foo podría cambiar si el espacio de nombres fuera redefinido y, por tanto, la firma digital no debería ser

igual si el espacio de nombres se redefine. Una solución rápida a este problema sería expandir el contenido del espacio de nombres en línea pero, desafortunadamente, existen mecanismos como XPath que consideran xx=”http://example.com” un espacio de nombres diferente a yy=”http://example.com”.

La diferencia fundamental entre la canonicalización inclusiva y exclusiva radica en el conjunto de espacio de nombres que muestran en su salida. Recordemos, que ambos mecanismos toman un documento XML como entrada y proporcionan la forma canónica (inclusiva o exclusiva) de ese documento en su salida. La canonicalizacion inclusiva copia todas las declaraciones de los espacios de nombres de los datos realmente firmados y de sus antecesores, incluso si estos se encuentran fuera del alcance de la firma. Además, también copia los atributos ‘xml:’ como por ejemplo ‘xml:lang’ o ‘xml:base’. De esta forma, esta transformación garantiza que todas las declaraciones de las que se pudieran hacer uso se encontrarán especificadas sin ambigüedad. Sin embargo, el problema surge cuando movemos los datos XML firmados de un documento a otro que posee otras declaraciones. En este caso la firma dejará de ser válida ya que cuando esta sea verificada se tendrá en cuenta otro contexto (aquel en el que se encuentran contenidos los datos) y por tanto fallará. El método de normalización exclusivo trata de analizar qué espacios de nombres están siendo realmente utilizados en el objeto de datos que será firmado y lo procesa de forma que en la salida sólo se encuentren los espacios de nombres “visiblemente” utilizados (aquellos que sean explícitamente referenciados en el objeto de datos XML a firmar). Este

mecanismo no busca definiciones de espacios de nombres en los valores de los atributos (<ppp:datos id=”foo:idpropio”) o de los elementos (<ppp:datos>foo:clave</ppp:datos>). La canonicalización exclusiva es útil cuando tenemos un documento XML firmado que deseamos insertar dentro de otros documentos XML sin que la firma se rompa. Un buen ejemplo es una afirmación SAML que pudiera ser insertada como un token XML en la cabecera de seguridad de varios mensajes SOAP (de hecho SAML permite utilizar el elemento *ds:Signature* como parte de la propia especificación).

Firmando los mensajes

La cabecera de seguridad Security definida por esta especificación se puede utilizar para transportar firmas digitales (conformes con la especificación XML Digital Signature). Esta cabecera Security incluida dentro de un sobre SOAP permitiría firmar uno o más elementos contenidos en su interior.

Para añadir una firma a un sobre SOAP mediante un bloque de cabecera *Security* se deberá agregar como elemento hijo inmediato al elemento *Security* un elemento *Signature* tal y como se encuentra definido por la especificación XML Digital Signature.

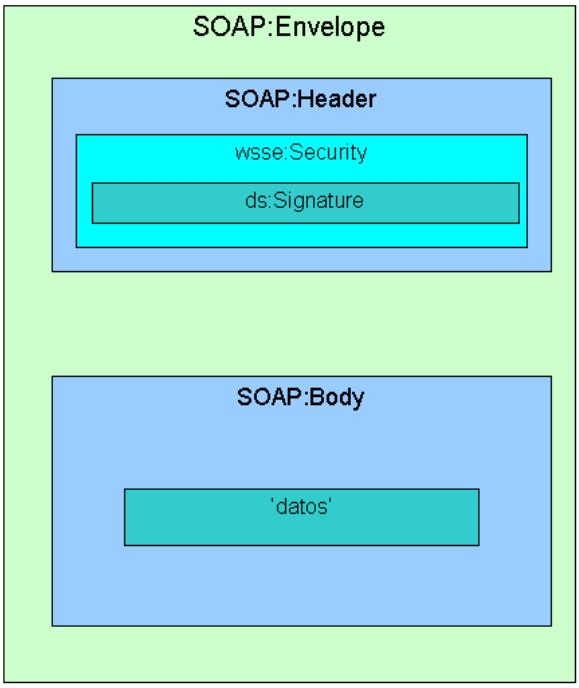

Ilustración 13. Ubicación del elemento ds:Signature dentro de la cabecera de seguridad wsse:Security.

Todos los elementos *Reference* incluidos en el elemento *SignedInfo* del elemento *Signature* deberán referenciar objetos de datos XML incluidos dentro del sobre SOAP. La especificación recomienda la utilización del algoritmo de canonicalización exclusivo para evitar problemas con las modificaciones que pudiera sufrir un mensaje SOAP en su viaje a través de intermediarios SOAP que lo modifiquen. Un simple cambio en la cabecera SOAP podría hacer que la firma incluida dentro del bloque de cabecera *Security* se rompiera cuando, en realidad, esto no debiera haber ocurrido.

Firmando de tokens de seguridad

A menudo se puede querer firmar tokens de seguridad (p.e.: una clave utilizada para cifrar los datos de un mensaje). La especificación XML Digital Signature define una serie de mecanismos como el uso de identificadores, URIs o expresiones XPath para referenciar los objetos de datos, en este caso los tokens de seguridad, que son firmados.

El problema es que pueden existir tokens de seguridad que no admitan ninguno de estos formatos de referencia y, por tanto, no pueden ser referenciados por ningún elemento *Reference* de una firma. La especificación en estudio permite que los tokens de seguridad dispongan de su propio mecanismo de referencia mediante la definición en su correspondiente perfil de token de seguridad de la extensión oportuna del elemento *SecurityTokenReference*. Como ya hemos visto anteriormente, este elemento ofrece un mecanismo uniforme que permite que todo tipo de token sea referenciable.

En el contexto de las firmas digitales, la especificación WS-Security define un nuevo mecanismo de transformación para XML Digital Signature denominado 'STR Derefence Transform'. Recordar que en las firmas digitales las transformaciones se aplican sobre los elementos que son referenciados para ser firmados (referenciados mediante el elemento *Reference*). El resultado de aplicar la transformación sobre el elemento referenciado se utiliza como entrada para una función resumen que proporcione un código de autenticidad de los datos. Esta nueva transformación está especificada por el fragmento de URI *'#STR-*

Transform' y cuando es aplicada sobre un elemento *SecurityTokenReference* significa que la salida será el token referenciado por el elemento *SecurityTokenReference*.

Como resumen del modelo de procesamiento de esta transformación, el procedimiento consiste en sacar, sin realizar alteración alguna, la información directamente a la salida de la transformación mientras el elemento encontrado no sea *SecurityTokenReference*. Cuando es un elemento de este tipo, el procedimiento, utilizando el criterio y las reglas definidos por este elemento, tratará de obtener el token referenciado y lo llevará directamente a la salida de la transformación. En el caso de que la transformación no pudiera deshacer la referencia a cierto token de seguridad el proceso lanzaría un error.

```
<?xml version="1.0" encoding="utf-8" ?>
 <S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="..."
       xmlns:ds="...">
    <S11:Header>
       <wsse:Security>
          <wsse:BinarySecurityToken ValueType="...#X509v3"
               EncodingType="...#Base64Binary"
               wsu:Id="X509Token">MIIEZzCCA9CgAwIBAgIQEmtJZc0rqrKh
               5i...
          </wsse:BinarySecurityToken>
          <ds:Signature>
             <ds:SignedInfo>
                <ds:CanonicalizationMethod
                   Algorithm="http://www.w3.org/2001/10/xml-exc-
                   c14n#" />
```

```
                    <ds:SignatureMethod
                        Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
                        sha1" />
                    <ds:Reference URI="#myBody">
                        <ds:Transforms>
                            <ds:Transform
                                Algorithm="http://www.w3.org/2001/10/xml-exc-
                                c14n#" />
                        </ds:Transforms>
                        <ds:DigestMethod
                            Algorithm="http://www.w3.org/2000/09/xmldsig#s
                            ha1" />
                        <ds:DigestValue>EULddytSo1...</ds:DigestValue>
                    </ds:Reference>
                </ds:SignedInfo>
                <ds:SignatureValue>BL8jdfToEb1l/vXcMZNNjPOV...</ds:SignatureV
            alue>
                <ds:KeyInfo>
                    <wsse:SecurityTokenReference>
                        <wsse:Reference URI="#X509Token" />
                    </wsse:SecurityTokenReference>
                </ds:KeyInfo>
            </ds:Signature>
        </wsse:Security>
    </S11:Header>
    <S11:Body wsu:Id="myBody">
        <tru:StockSymbol
            xmlns:tru="http://www.fabrikam123.com/payloads">QQQ</tru:
            StockSymbol>
    </S11:Body>
</S11:Envelope>
```

En este ejemplo se muestra un mensaje SOAP conforme a la versión 1.1 que incluye un bloque de cabecera *Security*. Este bloque de cabecera se compone de dos elementos: un token de seguridad binario que representa un certificado X509v3 y una firma digital aplicada sobre el contenido del cuerpo del mensaje SOAP (S11:Body). El token de seguridad representa el certificado que vincula una clave pública con la identidad del posesor de la clave. Se ha utilizado la clave privada para realizar la firma y debe ser utilizada la clave pública contenida en el certificado para poder llevar a cabo su verificación. En este caso el elemento *ds:KeyInfo* hijo del elemento *ds:Signature* contiene directamente el material de clave que permite verificar la firma digital. Otro contenido podría el nombre distinguido (DN) del posesor de la clave de forma que el receptor pudiera encontrar el certificado (con su clave pública correspondiente) en un servicio de directorio X.500. Otro contenido podría ser por ejemplo una URI que apuntara al contenido del certificado.

6.1.13 Cifrado de sub-elementos

WS-Security permite el cifrado de cualquier combinación de bloques del cuerpo, de la cabecera o de cualquiera de estas sub-estructuras, mediante:

- Una clave simétrica compartida por el emisor y el receptor.
- Una clave transportada en el mensaje de manera cifrada.

Con el fin de conseguir esta flexibilidad WS-Security se apoya en el estándar *XML Encryption*. Cuando un emisor o un intermediario cifra porciones de un mensaje SOAP mediante *XML Encryption* antepondrá un sub-elemento al bloque de cabecera *Security*. El proceso combinado de

cifrar porciones de un mensaje y agregar uno de estos sub-elementos que hacen referencia a las porciones cifradas se denominará *paso de cifrado* de aquí en adelante. El sub-elemento debería contener la suficiente información para que el receptor pueda identificar cuáles son las porciones del mensaje que deben ser descifradas por él.

Elemento xenc:ReferenceList

Cuando se cifran elementos o contenidos de elementos dentro de un sobre SOAP, el elemento *xenc:ReferenceList*, procedente de la especificación *XML Encryption*, podría utilizarse para crear un manifiesto de las partes que han sido cifradas, representadas por elementos *xenc:EncryptedData* dentro del sobre SOAP. El elemento *xenc:ReferenceList* contendrá un conjunto de referencias a la información cifrada utilizando el elemento *xenc:DataReference*. Un elemento, o el contenido de un elemento, que va a ser cifrado en un *paso de cifrado* debe ser reemplazado por el correspondiente *xenc:EncryptedData* de acuerdo a lo estipulado en la especificación *XML Encryption*. Todos los elementos *xenc:EncryptedData* creados por este *paso de cifrado* deberían estar listados en elementos *xenc:DataReference* contenidos dentro de un elemento *xenc:ReferenceList*. Aunque en *XML Encryption* el elemento *xenc:ReferenceList* tiene como propósito inicial ser utilizado dentro de un elemento *xenc:EncryptedKey* (lo cual implica que todos los elementos *xenc:EncryptedData* referenciados están cifrados por la misma clave), la especificación WS-Security permite que los elementos *xenc:EncryptedData* referenciados por el mismo *xenc:ReferenceList* puedan estar cifrados por claves distintas. Cada clave de cifrado puede

especificarse mediante el elemento *ds:KeyInfo* dentro un elemento *xenc:EncryptedData* independiente. Recordar de la especificación *XML Encryption* que un elemento *xenc:EncryptedData* contiene un elemento *xenc:EncryptionMethod* que indica el algoritmo de cifrado utilizado, un elemento *ds:KeyInfo* con la información de la clave utilizado en el cifrado, un elemento *xenc:CipherData* que, o bien contiene un elemento *xenc:CipherValue* con el grupo de octetos con los datos cifrados, o bien contiene un elemento *xenc:CipherReference* que referencia la ubicación de los octetos con los datos cifrados. Una situación típica donde el sub-elemento *xenc:ReferenceList* es útil es aquella en la que el emisor y el receptor utilizan un clave secreta compartida. El siguiente fragmento ilustra el uso de este sub-elemento:

```
<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
        xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext"
        xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
  <S:Header>
    <wsse:Security>
      <xenc:ReferenceList>
        <xenc:DataReference URI="#bodyID" />
      </xenc:ReferenceList>
    </wsse:Security>
  </S:Header>
  <S:Body>
    <xenc:EncryptedData Id="bodyID">
      <ds:KeyInfo>
        <ds:KeyName>CN=Hiroshi Maruyama, C=JP</ds:KeyName>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>...</xenc:CipherValue>
      </xenc:CipherData>
```

```
      </xenc:EncryptedData>
   </S:Body>
</S:Envelope>
```

Si observamos la cabecera del mensaje SOAP, nos encontramos con un único nodo hijo en forma de un bloque de cabecera WS-Security. El contenido de este elemento refleja una lista de referencias (en negrita) a porciones de datos cifrados. En este caso la referencia apunta al elemento XML con identificador *bodyID*, que es un elemento *xenc:EncryptedData* que contiene dos nodos hijos. El primero representa los datos necesarios para que el receptor pueda obtener la clave secreta a utilizar para poder descifrar los datos (*ds:KeyInfo*) y el segundo representa los propios datos cifrados mediante el elemento *xenc:CipherData.*

Elemento xenc:EncryptedKey

Cuando el *paso de cifrado* implica el cifrado de elementos o contenido de elementos dentro de un sobre SOAP con una clave, que a su vez debe ser cifrada por la clave de receptor e incrustada en el mensaje, se podría hacer uso del elemento *xenc:EncryptedKey*, descrito en *XML Encyption*, para transportar tal clave como una clave cifrada (esto es útil por ejemplo cuando una parte desea enviar una clave simétrica de sesión a la otra parte cifrándola con la clave pública del receptor). Este sub-elemento debería tener un manifiesto, esto es, un elemento *xenc:ReferenceList*, con el fin de que el receptor conozca las porciones que deben ser descifradas con esta clave (si es que existe alguna). Un elemento o el contenido de un elemento que debe ser cifrado por este *paso de cifrado* debe ser

reemplazado por el correspondiente elemento *xenc:EncryptedData* de acuerdo a lo indicado en la especificación *XML Encryption*. Todos los elementos *xenc:EncryptedData* creados por este *paso de cifrado* deberían encontrarse listados en el elemento *xenc:ReferenceList* dentro de este sub-elemento. Este constructor es útil cuando el cifrado es realizado por una clave simétrica generada aleatoriamente que es a su vez cifrada por la clave pública del receptor (proceso seguido por ejemplo en el protocolo SSL). El siguiente ejemplo ilustra el uso de este elemento:

```
<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
        xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext"
        xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
  <S:Header>
    <wsse:Security>
      <xenc:EncryptedKey>
        <xenc:EncryptionMethod Algorithm="..." />
        <ds:KeyInfo>
          <ds:KeyName>CN=Hiroshi Maruyama, C=JP</ds:KeyName>
        </ds:KeyInfo>
        <xenc:CipherData>
          <xenc:CipherValue>CLAVE SIMETRICA CIFRADA
          </xenc:CipherValue>
        </xenc:CipherData>
        <xenc:ReferenceList>
          <xenc:DataReference URI="#bodyID" />
        </xenc:ReferenceList>
      </xenc:EncryptedKey>
    </wsse:Security>
  </S:Header>
  <S:Body>
    <xenc:EncryptedData Id="bodyID">
      <ds:KeyInfo>
```

```
            <ds:KeyName>CN=Hiroshi Maruyama, C=JP</ds:KeyName>
        </ds:KeyInfo>
        <xenc:CipherData>
            <xenc:CipherValue>...</xenc:CipherValue>
        </xenc:CipherData>
    </xenc:EncryptedData>
  </S:Body>
</S:Envelope>
```

En este ejemplo vemos cómo el emisor envía una clave cifrada dentro del elemento *xenc:EncryptedKey*. Esa clave ha sido utilizada para cifrar los datos referenciados mediante el elemento *xenc:ReferenceList* y ha sido cifrada, pudiendo ser descifrada a partir de la clave cuya información asociada se indica en el elemento *ds:KeyName* incluido dentro del elemento *ds:KeyInfo.*

Elemento xenc:EncryptedData

En algunos casos la información relacionada con la seguridad se proporciona en una forma puramente cifrada. Anexos con formatos distintos a XML también pueden ser cifrados. El elemento *xenc:EncryptedData* definido en la especificación *XML Encryption* puede utilizarse en estos escenarios. Para cada parte del anexo a cifrar, es necesario un *paso de cifrado*; esto es, para cada anexo cifrado se debe añadir un sub-elemento *xenc:EncryptedData* siguiendo las siguientes reglas (subrayar que los pasos 2-4 se aplican solamente si los anexos son tipos MIME).

1. Los contenidos del anexo debe ser reemplazado por el octeto cifrado.

2. La parte MIME reemplazada debe tener el tipo de medio *application/octet-stream*.
3. El tipo de medio original del anexo debe ser declarado en el atributo *Mime Type* del elemento *xenc:EncryptedData.*
4. La parte MIME cifrada debe ser referenciada por un elemento *xenc:CipherReference* con una URI que apunte a la parte MIME con *cid:* como el componente de esquema de la URI.

El siguiente fragmento ilustra el uso de este elemento:

```
<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope"
      xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
      xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext"
      xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
   <S:Header>
      <wsse:Security>
         <xenc:EncryptedData MimeType="image/png">
            <xenc:EncryptionMethod Algorithm="foo:bar" />
            <ds:KeyInfo>
                  <ds:KeyName>CN=Hiroshi Maruyama, C=JP</ds:KeyName>
            </ds:KeyInfo>
            <xenc:CipherData>
                  <xenc:CipherReference URI="cid:image" />
              </xenc:CipherData>
           </xenc:EncryptedData>
        </wsse:Security>
     </S:Header>
     <S:Body />
</S:Envelope>
```

6.1.14 Reglas de procesamiento

Las partes cifradas o los anexos a los mensajes SOAP que utilicen alguno de los sub-elementos definidos anteriormente deben ser conformes con la especificación *XML Encryption*. Un sobre SOAP debe seguir siendo un sobre SOAP válido tras el *paso de cifrado*. El creador del mensaje **no**

debe cifrar los elemento *SOAP:Envelope*, *SOAP:Header* o *SOAP:Body* pero podría cifrar los elementos hijos de alguno de estos dos últimos. Se podrían agregar múltiples *pasos de cifrado* en un único bloque de cabecera *Security* en aquellos casos en los que estén destinados al mismo recipiente.

Cuando un elemento o el contenido de un elemento dentro de un sobre SOAP debe ser cifrado, deberá ser reemplazado por un elemento *xenc:EncryptedData* tal y como se define en la especificación *XML Encryption* y, además, deberá ser referenciado desde un elemento *xenc:ReferenceList* creado por este *paso de cifrado*.

Esta especificación permite colocar el flujo de octetos cifrados en un anexo. Por ejemplo, si un elemento *xenc:EncryptedData*, que aparece dentro de un elemento *SOAP:Body*, tiene un elemento *xenc:CipherReference* que referencia un anexo, entonces el flujo de octetos descifrados reemplazará el elemento *xenc:EncryptedData*. Sin embargo, si el elemento *xenc:EncryptedData* está colocado dentro de un bloque de cabecera *Security* y él referencia un anexo, entonces el flujo de octetos descifrados debe reemplazar el flujo de octetos cifrados en el anexo.

Cifrado

Los pasos generales para crear y cifrar mensajes SOAP conformes con la especificación WS-Security se muestran a continuación (subrayar que el uso del elemento *xenc:ReferenceList* es RECOMENDADO):

1. Crear un nuevo sobre SOAP.
2. Crear una cabecera de seguridad *wsse:Security.*
3. Crear un sub-elemento *xenc:ReferenceList*, un sub-elemento *xenc:EncryptedKey* o un sub-elemento *xenc:EncryptedData* en el bloque de cabecera *Security* dependiendo del tipo de cifrado.
4. Localizar los elementos de datos que van a ser cifrados.
5. Cifrar los elementos de datos de la siguiente manera: Para cada elemento XML o contenido del elemento dentro del sobre SOAP destino, cifrarlo de acuerdo a las reglas de procesamiento definidas en la especificación *XML Encryption*. Cada elemento o contenido del elemento original seleccionado debe ser eliminado y reemplazado por el elemento resultante *xenc:EncryptedData.* Para un anexo, los contenidos deben ser reemplazados por los datos cifrados tal y como se ha descrito en el apartado anterior.
6. El elemento opcional *ds:KeyInfo* dentro del elemento *xenc:EncryptedKey* podría referenciar a otro elemento *ds:KeyInfo.* Destacar que si el cifrado está basado en un token de seguridad anexado, entonces el elemento *SecurityTokenReference* deberá se añadido al elemento *ds:KeyInfo* para facilitar su localización. Por ejemplo, si se utiliza una clave pública incluida dentro de un certificado X509v3 para realizar el cifrado de los datos, podríamos representar dicha clave mediante un token de seguridad ubicado en la URL *http://example.com/claves/clave1*.

Como elemento hijo del elemento *ds:KeyInfo* podríamos incluir un elemento *wsse:SecurityTokenReference* que referencie mediante una URI a dicho token. De esta forma el receptor podrá recuperar de esta ubicación la clave, descifrar la clave cifrada contenida en el mensaje y descifrar, finalmente, los datos cifrados.

7. Crear un elemento *xenc:DataReference* que referencie los elementos *xenc:EncryptedData* generados. Añadir el elemento creado *xenc:DataReference* al elemento *xenc:ReferenceList* incluido dentro de la cabecera de seguridad.
8. Copiar en la salida todos los datos no cifrados.

Descifrado

Cuando se recibe un sobre SOAP con entradas de cabeceras cifradas, para cada cabecera cifrada se debe seguir los siguientes pasos generales para su procesado:

1. Localizar los elementos *xenc:EncryptedData* que deben ser descifrados (posiblemente se encuentren dentro de un elemento *xenc:ReferenceList* como parte de un elemento hijo más del bloque de cabecera *Security*).
2. Descifrar como se indica a continuación: para cada elemento en el sobre SOAP destino, descifrarlo de acuerdo a las reglas de procesamiento descritas en la especificación *XML Encryption* y las reglas de procesamiento ya descritas con anterioridad en este mismo documento.

3. Si los datos descifrados son parte de un anexo y fueron utilizados tipos MIME, entonces debemos revisar el tipo MIME del anexo del tipo MIME original (si es que existe alguno).

Si el proceso de descifrado falla por alguna razón, las aplicaciones podrían informar del fallo al emisor utilizando el código de fallo que se define más adelante en este apartado.

6.1.15 Ejemplo Extendido

El siguiente ejemplo ilustra el uso de los elementos de seguridad, firma y cifrado. Para este ejemplo, usamos una transformación ficticia *"Tranformación de Enrutamiento"* que selecciona las cabeceras de enrutamiento inmutables junto con el cuerpo del mensaje:

```
<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope"
      xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
      xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext"
      xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
   <S:Header>
      <m:path xmlns:m="http://schemas.xmlsoap.org/rp/">
           <m:action>http://fabrikam123.com/getQuote</m:action>
           <m:to>http://fabrikam123.com/stocks</m:to>
           <m:from>mailto:johnsmith@fabrikam123.com</m:from>
           <m:id>uuid:84b9f5d0-33fb-4a81-b02b-5b760641c1d6</m:id>
      </m:path>
      <wsse:Security>
           <wsse:BinarySecurityToken ValueType="wsse:X509v3"
               Id="X509Token"
               EncodingType="wsse:Base64Binary">MIIEZzCCA9CgAwIBAgIQ
               EmtJZc0rqrKh5i...</wsse:BinarySecurityToken>
         <xenc:EncryptedKey>
              <xenc:EncryptionMethod
                   Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-
                   1_5" />
             <ds:KeyInfo>
                  <ds:KeyName>CN=Hiroshi Maruyama, C=JP</ds:KeyName>
             </ds:KeyInfo>
```

```
            <xenc:CipherData>
                <xenc:CipherValue>d2FpbmdvbGRfE0Im4byV0...</xenc:Ciphe
                rValue>
            </xenc:CipherData>
            <xenc:ReferenceList>
                <xenc:DataReference URI="#enc1" />
            </xenc:ReferenceList>
        </xenc:EncryptedKey>
        <ds:Signature>
            <ds:SignedInfo>
                <ds:CanonicalizationMethod
                    Algorithm="http://www.w3.org/2001/10/xml-exc-
                    c14n#" />
                <ds:SignatureMethod
                    Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
                    sha1" />
                <ds:Reference>
                 <ds:Transforms>
                      <ds:Transform Algorithm="http://...#RoutingTransform"
                          />
                      <ds:Transform
                          Algorithm="http://www.w3.org/2001/10/xml-exc-
                          c14n#" />
                    </ds:Transforms>
                    <ds:DigestMethod
                        Algorithm="http://www.w3.org/2000/09/xmldsig#s
                        ha1" />
                    <ds:DigestValue>LyLsF094hPi4wPU...</ds:DigestValue>
                </ds:Reference>
            </ds:SignedInfo>
            <ds:SignatureValue>Hp1ZkmFZ/2kQLXDJbchm5gK...</ds:Signatu
                reValue>
            <ds:KeyInfo>
              <wsse:SecurityTokenReference>
                    <wsse:Reference URI="#X509Token" />
                </wsse:SecurityTokenReference>
             </ds:KeyInfo>
        </ds:Signature>
    </wsse:Security>
</S:Header>

<S:Body>
  <xenc:EncryptedData
        Type="http://www.w3.org/2001/04/xmlenc#Element"
        Id="enc1">
        <xenc:EncryptionMethod
            Algorithm="http://www.w3.org/2001/04/xmlenc#3des-cbc"
            />
          <xenc:CipherData>
          <xenc:CipherValue>d2FpbmdvbGRfE0Im4byV0...</xenc:CipherVal
              ue>
        </xenc:CipherData>
    </xenc:EncryptedData>
```

```
    </S:Body>
</S:Envelope>
```

Revisemos algunas de las secciones claves de este ejemplo:

Dentro del apartado de cabecera SOAP nos encontramos con el bloque de cabecera *Security* que contiene toda la información relacionada con la seguridad del mensaje. En su interior vemos que se ha definido un token de seguridad que está asociado con el mensaje. Un token de seguridad es un conjunto de afirmaciones de seguridad realizadas por un sujeto. En este caso, el token representa un certificado X.509 que está codificado en Base64. Un certificado X509v3 vincula a un usuario con una clave pública de su posesión. Esta vinculación está firmada por una tercera parte de confianza comúnmente denominada autoridad de certificación. A continuación se especifica la clave utilizada para cifrar el mensaje (contenido del elemento *xenc:EncryptedKey).* Ya que la clave utilizada para el cifrado es simétrica, se pasa de forma cifrada. El algoritmo utilizado para cifrar la clave es *http://www.w3.org/2001/04/xmlenc#rsa-1_5* y el nombre de clave utilizada para cifrar la clave simétrica se describe en el fragmento:

```
<ds:KeyInfo>
        <ds:KeyName>CN=Hiroshi Maruyama, C=JP</ds:KeyName>
</ds:KeyInfo>
```

El nombre distinguido en formato X.500: *CN=Hiroshi Maruyama, C=JP* no está indicando la entrada en un servicio de directorio en el que se encuentra la clave pública contenida en el certificado (expresado en el

token de seguridad pasado) dado y utilizada para cifrar la clave simétrica. Probablemente el señor Hiroshi Mayurama sea el destinatario final del mensaje y la clave pública le pertenezca a él. Cuando le llegue el mensaje descifrará con su clave privada la clave simétrica, de esta forma el emisor sabe con certeza que sólo el señor Maruyama podrá descifrar la clave de sesión, y después podrá descifrar el contenido de las partes cifradas mediante la clave de sesión.

Posteriormente se refleja la forma cifrada real de la clave simétrica y se referencia el bloque cifrado en el mensaje que utiliza la clave simétrica:

```
<xenc:CipherData>
    <xenc:CipherValue>d2FpbmdvbGRfE0lm4byV0...</xenc:CipherValue>
</xenc:CipherData>
<xenc:ReferenceList>
   <xenc:DataReference URI="#enc1" />
</xenc:ReferenceList>
```

En este caso sólo se ha cifrado el cuerpo del mensaje (id=”enc1”). A continuación se abre un elemento de firma *ds:Signature*. En este ejemplo, la firma está basada en el certificado X.509. El fragmento que se incluye dentro del elemento *ds:SignedInfo* indica las partes que están siendo firmadas así como el algoritmo de normalización (elemento *ds:CanonicalizationMethod*) y el método de firma empleado (contenido en *ds:SignatureMethod* y en este caso RSA sobre SHA1).

El elemento *ds:SignatureValue* indica el valor de la firma y el elemento *ds:KeyInfo* indica la clave que fue utilizada para aplicar la firma. En este caso, es el certificado X.509 incluido en el mensaje:

```
<ds:KeyInfo>
   <wsse:SecurityTokenReference>
     <wsse:Reference URI="#X509Token" />
  </wsse:SecurityTokenReference>
</ds:KeyInfo>
```

Posteriormente se cierra el elemento *SOAP:Header* y le sigue el elemento *SOAP:Body* que contiene el cuerpo del mensaje. En su interior encontramos metadatos cifrados y la forma del cuerpo del mensaje tal y como se indica en *XML Encryption*.

6.1.16 Sellos de tiempo de seguridad

La semántica de seguridad debe ser "renovada" periódicamente entre la partes interlocutoras. Muchas veces, con el objetivo de evitar ataques de robo de sesión o de ataques de repetición, una de las partes interlocutoras inicia un proceso por el cual trata de refrescar la semántica de seguridad establecida al inicio de sesión. Por ejemplo, si dos partes acordaron una clave de sesión con la que intercambiarse mensajes de forma confidencial, una de ellas, transcurrido cierto tiempo, podría querer renovar esta clave de forma que renovaría su confianza en que la otra parte sigue siendo quién dijo ser.

La especificación WS-Security no proporciona mecanismos explícitos para llevar a cabo esta renovación de la semántica de la seguridad ni para realizar una re-sincronización de los tiempos entre las partes. Asume que el tiempo marcado en los mensajes es de confianza y no ofrece elementos para su sincronización. La especificación WS-Security representa los sellos de tiempo mediante tipos de datos *xsd:dateTime* definido por la

especificación XML Schema. Además, recomienda que los tiempos se encuentren en formato UTC. Si se utilizarán otros formatos para representar el tiempo, será necesario hacer uso del atributo *ValueType* (que describiremos más adelante). Otra recomendación más que realiza la especificación es que las partes no deberán confiar en otras aplicaciones que soporten resoluciones de tiempo más precisas que los milisegundos.

El elemento *wsu:Timestamp* (el espacio de nombres cuyo prefijo es wsu contiene declaraciones de tipos y elementos que representan utilidades y que son utilizados por diversas especificaciones) proporciona un mecanismo para expresar la creación y expiración de los tiempos de las semánticas de la seguridad en un mensaje.

El elemento *wsu:Timestamp* puede encontrarse como un elemento hijo de la cabecera de seguridad *wsse:Security* y sólo puede existir uno por cabecera o, lo que es lo mismo, por actor/rol SOAP. A continuación se presenta la sintaxis de este elemento:

```
<wsu:Timestamp wsu:Id="...">
  <wsu:Created ValueType="...">...</wsu:Created>
  <wsu:Expires ValueType="...">...</wsu:Expires>
  ...
</wsu:Timestamp>
```

Como podemos ver, una marca de tiempo puede definir un atributo que le permite tener un identificador único a ese sello de tiempo. Después, puede contener opcionalmente un elemento *wsu:Created* que representa el instante en el que las semánticas de seguridad fueron creadas. En el modelo de procesamiento SOAP, este instante es aquel en el que el

conjunto de información es serializada para llevar a cabo su transmisión. El tiempo de creación y de transmisión debería estar minimizado de forma que no difiera sustancialmente. El otro elemento hijo posible es *wsu:Expires* que, como podemos imaginarnos, representa el instante en el tiempo en el que la cabecera SOAP de seguridad *Security* y las semánticas de seguridad en ella contenidas pueden considerarse expiradas. Tras haber expirado, el solicitante confirma que las semánticas de seguridad ya no son válidas, y el receptor debería, al menos así lo recomienda enérgicamente la especificación, descartar cualquier mensaje con la cabecera expirada. El receptor podría utilizar un mensaje de error (*wsu:MessageExpired*) para comunicar al emisor la expiración de la cabecera de seguridad. La especificación indica que un servicio debería emitir un Fallo SOAP indicando que la semántica de seguridad ha caducado. Como medida extensibilidad, el elemento *wsu:Timestamp* podría tener cualquier otro elemento hijo o definir cualquier otro tipo de atributo.

El tiempo de referencia a utilizar para determinar si una cabecera *Security* ha expirado es el utilizado por el solicitante. Con el fin de evaluar el tiempo de expiración, los receptores pueden considerar que el reloj de los solicitantes pudiera estar de-sincronizado con el suyo. El receptor, por tanto, debe realizar una afirmación que indique el nivel de confianza aplicado al reloj del solicitante, puesto que el receptor es el que debe evaluar si el tiempo de expiración se ha superado o no en función al tiempo de reloj del solicitante.

El siguiente fragmento XML ilustra el uso del elemento en cuestión:

```
<S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="...">
  <S11:Header>
    <wsse:Security>
      <wsu:Timestamp wsu:Id="timestamp">
          <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
          <wsu:Expires>2001-10-13T09:00:00Z</wsu:Expires>
        </wsu:Timestamp>
      </wsse:Security>
      ...
    </S11:Header>
    <S11:Body>...</S11:Body>
</S11:Envelope>
```

6.1.17 Ejemplo extendido

En esta sección vamos a estudiar en detalle un ejemplo extendido que utiliza gran parte de los elementos y mecanismos definidos en la especificación. Este ejemplo está extraído de la especificación.

En primer lugar definamos los requisitos de seguridad que queremos utilizar para llevar a cabo el envío de un mensaje conforme a esta especificación. El emisor desea enviar el siguiente mensaje al receptor:

```
<foo:invitacion tipo="boda" wsu:id="1332">
    <foo:dia>22/5/2004</foo:dia>
    <foo:lugar>Madrid</foo:lugar>
    <foo:iglesia>Los Jeronimos</foo:iglesia>
   <foo:comentarios>Se ruega confirmar
        asistencia</foo:comentarios>
 </foo:invitacion>
```

El emisor, Pedro, desea enviar este mensaje de manera confidencial a Ana ya que desea invitarla a su boda. Con el objeto de que una atacante no pueda leer el mensaje, se desea cifrar este mensaje. Además, y con el

objeto de que ambos interlocutores sepan que el otro extremo es quién dice ser se deberán incluir mecanismos de seguridad que los identifique. Apliquemos pues los mecanismos expuestos en la especificación para crear un mensaje SOAP que contenga una cabecera de seguridad *Security* que albergue las semánticas de seguridad aplicadas en el mensaje. Estas semánticas consistirán en cifrar el mensaje de invitación mediante una clave de sesión acordada entre las partes cifrada a su vez mediante la clave pública de Ana. De esta forma sólo Ana podrá descifrar la clave de sesión mediante su clave privada. Además, Pedro sabe con total certeza que sólo Ana será capaz de descifrar el mensaje (ya que es la única poseedora de su clave privada).

Pedro firmará el mensaje con su clave privada de manera que Ana pueda saber con certeza que es Pedro el que realmente le envía el mensaje.

Con estas medidas de seguridad garantizamos la autenticación de las partes, la integridad del mensaje, y su confidencialidad.

La cabecera de seguridad del mensaje se compone de un *timestamp*, un token de seguridad binario, una clave cifrada (utilizada para cifrar el cuerpo del mensaje) y una firma digital aplicada sobre el *timestamp*.

Con el primer elemento se conoce que la firma fue realizada en cierto instante de tiempo asociado con la generación del mensaje. Esto es equivalente a cuando firmamos un documento en papel y lo hacemos sobre la fecha en la que el documento fue firmado.

El tercer elemento, *xenc:EncryptedKey* permite transportar la clave con la que se encuentra cifrado el cuerpo del documento. Su elemento hijo *xenc:ReferenceList* se constituye como un manifiesto de las partes que han sido cifradas con la clave cifrada y el elemento *ds:KeyInfo* contiene la información necesaria, en forma de identificador, de la clave utilizada para cifrar la clave simétrica. En este caso la parte cifrada con dicha clave es el cuerpo del mensaje SOAP cuyo identificador de fragmento es '#enc1'.

El cuarto componente de la cabecera de seguridad es una firma digital acorde a la especificación *W3C XML Digital Signature*. Dicha firma digital se ha aplicado sobre el *timestamp* mencionado y sobre el cuerpo del mensaje (como se verá en el documento XML final existen dos elementos *ds:Reference* uno para cada objeto de datos a firmar).

La estructura general que tendrá el mensaje es la siguiente:

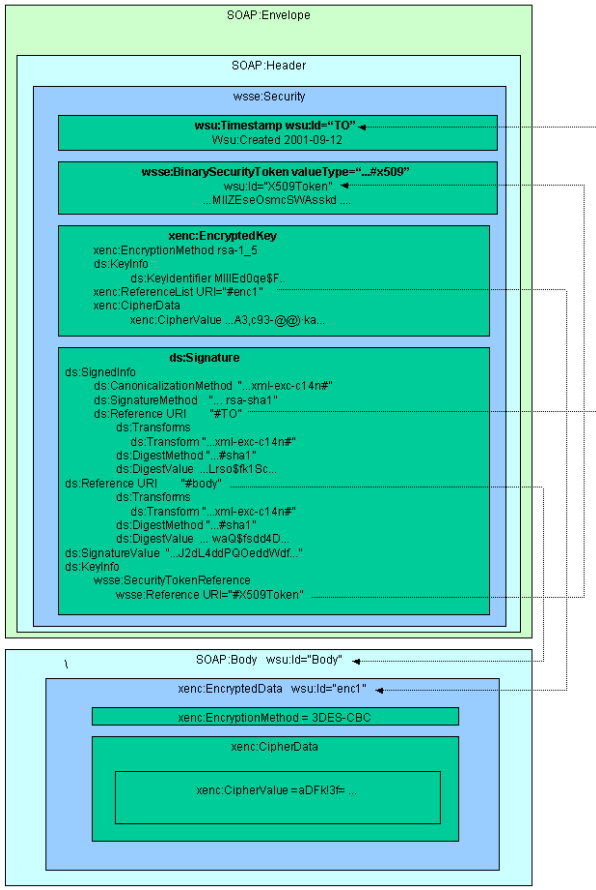

Ilustración 14. Estructura del ejemplo extendido de un mensaje WS-Security.

Y el documento XML correspondiente es:

```
<?xml version="1.0" encoding="utf-8" ?>
<S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="..."
    xmlns:xenc="..." xmlns:ds="...">
  <S11:Header>
    <wsse:Security>
      <wsu:Timestamp wsu:Id="T0">
          <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
      </wsu:Timestamp>
      <wsse:BinarySecurityToken ValueType="...#X509v3"
          wsu:Id="X509Token"
          EncodingType="...#Base64Binary">MIIEZzCCA9CgAwIBAgIQE
          mtJZc0rqrKh5i...</wsse:BinarySecurityToken>

       <xenc:EncryptedKey>
          <xenc:EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
          />
        <ds:KeyInfo>
            <wsse:KeyIdentifier EncodingType="...#Base64Binary"
                ValueType="...#X509v3">MIGfMa0GCSq...</wsse:KeyIdent
                ifier>
          </ds:KeyInfo>
          <xenc:CipherData>

              <xenc:CipherValue>d2FpbmdvbGRfE0lm4byV0...</xenc:Ci
              pherValue>
          </xenc:CipherData>
         <xenc:ReferenceList>
            <xenc:DataReference URI="#enc1" />
          </xenc:ReferenceList>
        </xenc:EncryptedKey>
        <ds:Signature>
          <ds:SignedInfo>
            <ds:CanonicalizationMethod
                Algorithm="http://www.w3.org/2001/10/xml-exc-
                c14n#" />
            <ds:SignatureMethod
                Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
                sha1" />
            <ds:Reference URI="#T0">
              <ds:Transforms>
                <ds:Transform
                    Algorithm="http://www.w3.org/2001/10/xml-exc-
                    c14n#" />
                </ds:Transforms>
                <ds:DigestMethod
                Algorithm="http://www.w3.org/2000/09/xmldsig#sha1
                " />
               <ds:DigestValue>LyLsF094hPi4wPU...</ds:DigestValue>
```

```
                </ds:Reference>
                <ds:Reference URI="#body">
                  <ds:Transforms>
                    <ds:Transform
                      Algorithm="http://www.w3.org/2001/10/xml-exc-
                      c14n#" />
                  </ds:Transforms>
                  <ds:DigestMethod
                    Algorithm="http://www.w3.org/2000/09/xmldsig#s
                    ha1" />
                  <ds:DigestValue>LyLsF094hPi4wPU...</ds:DigestValue>
                </ds:Reference>
              </ds:SignedInfo>
              <ds:SignatureValue>Hp1ZkmFZ/2kQLXDJbchm5gK...</ds:Signa
              tureValue>
              <ds:KeyInfo>
                <wsse:SecurityTokenReference>
                  <wsse:Reference URI="#X509Token" />
                </wsse:SecurityTokenReference>
              </ds:KeyInfo>
            </ds:Signature>
          </wsse:Security>
        </S11:Header>
        <S11:Body wsu:Id="body">
          <xenc:EncryptedData
            Type="http://www.w3.org/2001/04/xmlenc#Element"
            wsu:Id="enc1">
            <xenc:EncryptionMethod
              Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-
              cbc" />
            <xenc:CipherData>
              <xenc:CipherValue>d2FpbmdvbGRfE0lm4byV0...</xenc:CipherValu
              e>
            </xenc:CipherData>
          </xenc:EncryptedData>
        </S11:Body>
      </S11:Envelope>
```

6.1.18 Manejo de errores

Existen muchas circunstancias donde puede ocurrir un error mientras se procesa la información de seguridad. Por ejemplo:

- El algoritmo de cifrado o firma, o el token de seguridad no es válido o no es soportado por la aplicación que lo procesa.

- El token de seguridad no se puede autenticar, no está autenticado, o no es válido.
- Firma no válida.
- Fallo en el descifrado.
- Token de seguridad referenciado no disponible.
- Que el espacio de nombres de los datos de negocio incluidos en el cuerpo del mensaje no se encuentre disponible.

Si cierto de consumidor de un mensaje no puede llevar a cabo su operativa normal debido al contenido de la cabecera de seguridad deberá señalar un fallo tal y como se indica en la especificación SOAP. Las siguientes tablas contienen los códigos de seguridad predefinidos por la especificación.

Las clases de errores “no soportados” son:

Error	**Código de error**
Se proporcionó un token de seguridad que la aplicación no soporta.	wsse:UnsupportedSecurityToken
Se proporcionó una firma o un algoritmo de cifrado que la aplicación no soporta.	wsse:UnsupportedAlgorithm

Tabla 2. Errores manejados por la especificación WS-Security.

Las clases de error “fallo” son:

Error	Código de error
Se descubrió un error procesando la cabecera <Security>.	wsse:InvalidSecurity
Se proporcionó un token de seguridad no válido.	wsse:InvalidSecurityToken
El token de seguridad no pudo ser autenticado o autorizado.	wsse:FailedAuthentication
La firma o el desciframiento fueron erróneos.	wsse:FailedCheck
La referencia al token de seguridad no puede ser recuperada.	wsse:SecurityTokenUnavailable

Tabla 3. Fallos soportados por la especificación WS-Security.

6.1.19 Consideraciones de Seguridad

Esta especificación que precisa un marco de trabajo para la transferencia de mensajes SOAP utilizando mecanismos de seguridad no es segura por sí misma. Para que su aplicación sea segura se deben tener en cuenta ciertas consideraciones. Entre los puntos de seguridad a considerar están:

- Garantía de la “frescura” de las semánticas de la seguridad.
- Utilización adecuada de las firmas digitales y de los mecanismos de cifrado. No vale de nada cifrar o firmar si la clave utilizada está comprometida.

- Protección de los tokens de seguridad (integridad).
- Verificación de los certificados (incluyendo temas como su revocación). Un sistema que recibe un certificado debería verificar que dicho certificado no se encuentra en la lista de revocación de su autoridad emisora.
- El tradicional “agujero” de seguridad de las claves débiles.
- El uso de aleatoriedad (o pseudo-aleatoriedad fuerte).
- Interacción entre los mecanismos de seguridad que implementan este estándar y otros componentes del sistema.
- Ataques de “hombre-en-el-medio”.
- Ataques PKI.

Con el objetivo de detectar ataques de repetición, se recomienda que los mensajes incluyan elementos firmados digitalmente para permitir a los receptores detectar repeticiones de los mensajes cuando los mensajes son intercambiados en una red abierta. Estas firmas estar ubicadas en el propio mensaje o en alguna de las cabeceras definidas por otras extensiones SOAP. Cuatro posibles aproximaciones al problema son:

- Timestamp.
- Número de secuencia.
- Expiraciones.
- Correlación de mensajes.

Las firmas digitales necesitan ser comprendidas en el contexto de otros mecanismos de seguridad y las posibles amenazas deben estar completamente asimiladas. Las firmas digitales por sí solas no

proporcionan autenticación de mensaje. Uno puede registrar un mensaje firmado y re-enviarlo (en un ataque de repetición). Para prevenir este tipo de ataque, las firmas digitales deben ser combinadas con medios apropiados, como sellos de tiempo o números de secuencia, para asegurar que sólo existe una copia del mensaje.

Cuando las firmas digitales son utilizadas para verificar la identidad de la parte emisora, el emisor debe probar la posesión de la clave privada. Una forma de conseguir esto es utilizar un protocolo de tipo reto-respuesta. Este protocolo está fuera del alcance de este documento (ver WS-Trust (S. Anderson, J. Bohren, T. Boubez, M. Chanliau, G. Della-Libera, B. Dixon, P. Garg, E. Gravengaard, M. Gudgin, P. Hallam-Baker et al., 2004)).

Los clientes de un servicio deben utilizar firmas digitales para firmas tokens de seguridad que incluyan ya firmas digitales (u otros mecanismos de protección) para asegurar que no han sido alterados durante el tránsito.

También, y por último, en *XML Encryption*, destacaremos que la combinación de técnicas de firmado y cifrado son un elemento de datos común introduce algunas vulnerabilidades criptográficas. Por ejemplo, cifrar los datos que han sido firmados digitalmente, mientras se deja el valor de la firma en claro podría permitir ataques de *adivinación de texto plano*. Los diseñadores deben tomar especial precaución para no introducir este, u otro, tipo de vulnerabilidades.

6.1.20 Interoperabilidad

La especificación, basándose en la experiencia de interoperabilidad con esta y otras especificaciones, ofrece una lista de puntos a tener en cuenta con el objeto de que los desarrolladores de esta especificación no cometan ciertos errores o caigan en ciertas trampas desde el punto de vista de la interoperabilidad:

- Asegurarse una comprensión y aplicación correcta del algoritmo de identificación de claves.
- El elemento *xenc:EncryptedKey* contiene un atributo *Type* cuyo rango de posibles valores está predefinido, teniéndose que asegurar utilizar uno de estos valores.
- Los identificadores reconocidos por esta especificación (el atributo *wsu:Id* y los atributos *Id* locales en firmas digitales y elementos cifrados) deben ser utilizados apropiadamente.
- Los formatos de las horas.
- Incorrecta aplicación de las semánticas definidas por especificaciones como SOAP, WSDL o HTTP.

6.2 SAML

Organización	OASIS
Estado	Liberada
Versión	1.1 (3 Septiembre 2003)

6.2.1 Introducción

SAML (Farrell et al., 2003) es una especificación liberada por OASIS y su última versión es la 1.1 publicada el 3 de Septiembre del 2003.

SAML (Security Assertion Markup Language) es un marco de trabajo XML que permite el intercambio de información de seguridad entre servicios Web. Esta información de seguridad se materializa en forma de afirmaciones hechas por una autoridad SAML sobre un sujeto. Las afirmaciones SAML pueden contener tres tipos de información: autenticación, atributo, y decisión de autorización.

El sujeto de una afirmación es aquella entidad objeto de las afirmaciones realizadas por la autoridad SAML. En SAML se definen tres tipos lógicos de autoridades en función al tipo de declaraciones que emiten en sus afirmaciones: Autoridad de Autenticación, Autoridad de Atributos y Puntos de Decisión de Políticas.

Además, SAML define un protocolo petición/respuesta utilizado entre los sujetos y las autoridades SAML. Este protocolo se mapea con protocolos subyacentes (HTTP) mediante la definición normativa definida por un vínculo.

Uno de los principales propósitos de diseño de SAML es conseguir un modelo de autenticación centralizado Single Sign-On. Sin embargo, SAML puede ser utilizado para soportar diferentes estilos de SSO o para proporcionar seguridad sobre la carga de negocio de los mensajes SOAP. Los modos de utilizar SAML para cada uno de los diferentes escenarios se definen por la especificación mediante los perfiles.

Además, SAML se apoya en otras especificaciones como por ejemplo XML Digital Signature, para conseguir autenticidad, integridad y (posiblemente) no repudio, de cuyo proceso de estandarización es responsable el consorcio W3C.

6.2.2 Afirmaciones SAML

Las afirmaciones SAML son paquetes de información que contienen un conjunto de declaraciones realizadas por autoridades SAML.

Toda afirmación SAML contiene dos partes: una parte contiene información común a todos los tipos de afirmaciones y la otra contiene información particular para el tipo de afirmación.

A continuación se muestra un documento SAML:

```
<saml:Assertion AssertionID="2se8e/vaskfsdif=" Issuer="www.sts.com"
      IssueInstant="2002-06-19T16:58:33.173Z" xmlns:saml="saml">
   <saml:Conditions NotBefore="2002-06-19T16:53:33.173Z"
         NotOnOrAfter="2002-06-19T17:08:33.173Z" />
   <saml:AuthenticationStatement
         AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:X.509"
         AuthenticationInstant="2002-06-19T16:57:30.000Z">
      <saml:Subject>
         <saml:NameIdentifier
               NameQualifier="service.com">Client</saml:NameIdentifier>
         <saml:SubjectConfirmation>
            <saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:c
                  m:sender-vouches</saml:ConfirmationMethod>
         </saml:SubjectConfirmation>
      </saml:Subject>
   </saml:AuthenticationStatement>
   <ds:Signature xmlns:ds="ds" />
</saml:Assertion>
```

El elemento *Assertion* encapsula toda la información contenida dentro una afirmación. Cualquier documento que represente una afirmación SAML puede ser referenciado mediante el elemento *AssertionIDReference*.

Datos como la versión SAML en uso, un identificador único para la afirmación, la autoridad SAML que la creó y el instante de tiempo, basado en UTC, en el que fue creada son atributos **obligatorios** del elemento *Assertion* que describen la información común de cualquier afirmación SAML. Además, es posible agregar más datos opcionales como por ejemplo aquel que nos permite reflejar las condiciones (elemento *Conditions*) a tener en cuenta cuando se evalúa si la afirmación es o no válida, una firma digital (mediante el elemento *ds:Signature*) o información adicional (mediante el elemento *Advice*) que desee la autoridad SAML anexar en la afirmación.

A continuación se debe definir la información que contiene las declaraciones mediante uno o varios de entre los siguientes elementos:

- *Statement*: una declaración realizada mediante una extensión al esquema.
- *SubjectStatement*: una declaración del sujeto de la afirmación realizada mediante una extensión del esquema.
- *AuthenticationStatement*: una afirmación de autenticación.
- *AuthorizationDecisionStatement*: una afirmación que contiene la información relacionada con la decisión resultado de una evaluación de autorización.
- *AttributeStatement*: una afirmación que contiene información de los atributos del sujeto en favor del cual se está emitiendo la aseveración.

6.2.3 Condiciones de evaluación

Como ya se ha indicado, uno de los elementos opcionales que podemos incluir dentro del elemento *Assertion*, que recordemos es el elemento raíz de cualquier tipo de afirmación que queramos crear, es aquel que nos permite indicar bajo qué condiciones se debe evaluar la afirmación para garantizar su validez.

El elemento que nos permite reflejar esta información es *Conditions*. Este elemento puede incluir opcionalmente 2 atributos: *NotBefore* y *NotOnOrAfter*. Su uso permite, respectivamente, indicar el instante de tiempo a partir del cual la aseveración será válida y el instante de tiempo a partir del cual dejará de serlo. Ambos sellos de tiempo estarán en el

formato definido por el esquema de tipos de datos W3C XML Schema Datatypes y se deben especificar en UTC. Además de estos dos atributos, el elemento *Conditions* puede contener los siguientes elementos opcionales: *AudienceRestrictionCondition*, *DoNotCacheCondition*, *Condition*.

El elemento *AudienceRestrictionCondition*, que puede aparecer varias veces, permite indicar la audiencia para la que la aseveración está dirigida.

El elemento *DoNotCacheCondition*, que también puede ocurrir varias veces, indica al motor que evalúe la afirmación que no debe cachear el resultado de evaluar las condiciones teniendo que evaluarlas siempre en un futuro.

El elemento *Condition* se puede utilizar como punto de extensión para definiciones de condiciones personalizadas que extiendan el esquema XML de SAML.

6.2.4 Declaraciones SAML

Como ya hemos visto, el elemento *Assertion* contiene una serie de atributos obligatorios y elementos comunes a todos los tipos de afirmaciones más, los elementos particulares en forma de declaraciones.

Una afirmación puede contener una o varias declaraciones. A continuación haremos un estudio de los posibles tipos de declaraciones que podemos definir.

El elemento *Statement*, como ya se ha mencionado, es un posible punto de extensión que permite a otros tipos de aplicaciones utilizar el framework SAML para el intercambio de otros tipos de declaraciones además de los definidos por la especificación.

El elemento *SubjectStatement* es otro tipo de declaración que permite a las aplicaciones basadas en afirmaciones reutilizar el framework SAML. Esta declaración contiene un elemento *Subject* que permite a la autoridad SAML describir al sujeto de la afirmación. Si le echamos un vistazo al esquema XML de SAML el elemento *Subject* se define de dos formas: mediante una secuencia de un elemento *NameIdentifier* y un elemento opcional *SubjectConfirmation* o tan sólo mediante un elemento *SubjectConfirmation*.

Si una autoridad SAML incluye ambos elementos, *NameIdentifier* y *SubjectConfirmation*, estará permitiendo a una entidad SAML que posteriormente reciba la afirmación realizar la confirmación de la identidad del sujeto incluida en el elemento *SubjectConfirmation* y así confiar en que la identidad que le presenta la afirmación es aquella reflejada en el elemento *NameIdentifier*.

El elemento *NameIdentifier* es una cadena de caracteres simple que indica el nombre del sujeto y que además puede contener dos atributos, *NameQualifier* y *Format*, ambos también cadenas de texto. El primero de estos atributos opcionales permite reflejar el dominio administrativo o de seguridad en el que está definido el contenido del elemento *NameIdentifier*. Es decir, podríamos decir que el sujeto con nombre pepe@foo.com (*NameIdentifier*) pertenece al dominio de seguridad foo.com (*NameQualifier*). El siguiente fragmento representaría a dicho usuario referenciado dentro de una afirmación de autenticación:

```
<saml:Assertion ... >
    <saml:Conditions ... />
   <saml:AuthenticationStatement ... >
      <saml:Subject>
            <saml:NameIdentifier
            NameQualifier="foo.com">pepe</saml:NameIdentifier>
            ...
```

Otro atributo opcional del elemento *NameIdentifier* es *Format*. Este atributo es una URI que debe referenciar el formato en el que se encuentra el valor del elemento *NameIdentifier*.

Por su parte, el elemento *SubjectConfirmation*, especifica un sujeto utilizando información que permite autenticarlo. Tal y como está definido en el esquema XML, es de tipo complejo y contiene una secuencia de los tres elementos siguientes:

- *ConfirmationMethod*: deben existir una o más ocurrencias de este elemento. Es una referencia URI que identifica el mecanismo de confirmación que puede ser utilizado para autenticar al sujeto.

Actualmente, SAML define una serie de identificadores en uno de sus anexos para ciertos tipos de mecanismos de confirmación.

- *SubjectConfirmationData:* información de autenticación adicional que puede ser utilizada por un protocolo de autenticación específico.
- *ds:KeyInfo:* elemento importado del esquema definido por la especificación W3C XML Digital Signature y que sirve para contener información de una clave de la que el sujeto de la afirmación es propietario.

En resumen, dentro de un elemento *SubjectStatement* indicamos el sujeto de la afirmación mediante un elemento *Subject*. Este elemento puede contener un elemento *NameIdentifier*, que indica en texto el nombre del sujeto, y un elemento *SubjectConfirmation*, que contiene información que permite a cualquier servicio SAML autenticar al sujeto que le presenta una afirmación. Si aparece el elemento *NameIdentifier* puede o no aparecer el elemento *SubjectConfirmation*. También puede aparecer sólamente el elemento *SubjectConfirmation*. El elemento *NameIdentifier* puede tener asignado un atributo *NameQualifier* para indicar un dominio de seguridad en el que el nombre dado en *NameIdentifier* es válido y un atributo *Format* que indica su formato. Por su parte, el elemento *SubjectConfirmation* contiene uno o más elementos *ConfirmationMethod* (referencias URIs algunas predefinidas en la especificación SAML en el apartado 8) que referencia el mecanismo que cualquier servicio SAML debe utilizar para autenticar al sujeto. Además, *SubjectConfirmation* puede, opcionalmente, contener un elemento *SubjectConfirmationData*,

que contiene información adicional necesaria por el mecanismo de autenticación referenciado en *ConfirmationMethod*, elemento que, al menos, debe aparecer una vez. Por último, *SubjectConfirmation* puede, también opcionalmente, contener el elemento *ds:KeyInfo* ,importado del esquema definido por la especificación XML Digital Signature, para indicar una clave propiedad del sujeto de la afirmación.

Hasta ahora hemos estudiado dos de los posibles elementos declarativos que contempla la especificación: *Statement* y *SubjectStatement*. Pasemos a estudiar ahora el elemento que representa una declaración de autenticación *AuthenticationStatement*.

6.2.5 Afirmación de autenticación

Esta declaración, hecha por una autoridad SAML, afirma que la identidad del sujeto de la afirmación ha sido confirmada mediante algún mecanismo de autenticación. El tipo de este elemento es *AuthenticationStatementType* que extiende del tipo *SubjectStatementAbstractType*, el mismo que el de las declaraciones *SubjectStatement*, añadiendo una serie de atributos y elementos adicionales: *AuthenticaciontMethod*, *AuthenticationInstant*, *SubjectLocality* y *AuthorityBinding*.

AuthenticationMethod es un atributo obligatorio del elemento *AuthenticationStatement* y su valor es una URI que referencia el método de autenticación utilizado para confirmar la identidad del sujeto de la aseveración.

AuthenticationInstant es otro atributo obligatorio del elemento *AuthenticationStatement* que refleja el instante del tiempo en el que la autenticación se produjo. Este valor está codificado en UTC.

SubjectLocality es un elemento, que puede aparecer como mucho una vez, hijo opcional del elemento *AuthenticationStatement* que contiene información del dominio DNS o de la IP del sistema que fue autenticado mediante dos atributos opcionales, *DNSAddress* e *IPAddress* respectivamente. Tal y como advierte la especificación ambos atributos son fácilmente 'spoofed' por lo que sólo son valores informativos que no deben ser tenidos en cuenta para autenticar al sujeto.

AuthorityBinding es un elemento opcional hijo del elemento *AuthenticationStatement.* Puede aparecer cero o varias veces. Sirve para indicar a cualquier servicio SAML que procese un elemento *AuthenticationStatement* que existe una autoridad SAML que podría proporcionarle información adicional sobre el sujeto de la declaración. Una autoridad SAML puede anunciar su presencia sobre varios protocolos, en varias ubicaciones y como más de un tipo de autoridad mediante el envío de tantos elementos *AuthorityBinding* como necesite. Toda esta información sobre el tipo, protocolo y ubicación de la autoridad SAML se puede indicar mediante tres atributos obligatorios del elemento *AuthorityBinding*: *AuthorityKind*, *Location*, *Binding*. El primero es de tipo QName definido en el SchemaXML y los dos últimos son referencias URI.

Una vez estudiado el tipo de declaración *AuthenticationStatement* pasemos a analizar la declaración que permite reflejar atributos asignados al sujeto de la aseveración: *AttributeStatement*.

6.2.6 Afirmación de atributos

La declaración de tipo *AttributeStatement* se utiliza para relacionar ciertas propiedades o atributos con el sujeto indicado en la aseveración. Sirve para que una autoridad SAML puede emitir aseveraciones en las que indique que cierto sujeto tiene ciertos atributos de forma que toda entidad que tenga establecida una relación de confianza con la autoridad SAML tenga la certeza de que cierto sujeto tiene ciertas características. Normalmente, estas características serán una de las variables evaluadas por un Punto de Decisión de Autorización SAML a la hora de tomar una decisión de autorización de cierto sujeto a cierto recurso.

Analizando el esquema XML de SAML vemos como el elemento *AttributeStatement* es del tipo *AttributeStatementType* que, al igual que ocurre con el tipo *AuthenticacionStatementType* del elemento *AuthenticationStatement*, extiende el tipo *SubjectStatementType* agregando uno o varios elementos *Attribute* en su definición.

Como nos podemos imaginar, el elemento *Attribute* contiene la información de un atributo de un sujeto. Si nos fijamos en la definición del esquema XML de este elemento:

```
<element name="Attribute" type="saml:AttributeType" />
<complexType name="AttributeType">
```

```
<complexContent>
  <extension base="saml:AttributeDesignatorType">
    <sequence>
      <element ref="saml:AttributeValue" maxOccurs="unbounded" />
    </sequence>
  </extension>
</complexContent>
</complexType>
```

Podemos observar como la definición del tipo de un atributo, *AttributeType*, extiende de la definición del tipo *AttributeDesignatorType*. Este tipo está asignado a un elemento *AttributeDesignator* el cual representa el nombre de un atributo dentro de un espacio de nombres de atributos. Normalmente, este elemento es utilizado en una petición del valor de un atributo y debe especificar dos atributos obligatorios. El primer atributo es *AttributeName*, es de tipo cadena de texto y representa el nombre del atributo. El segundo es *AttributeNamespace*, es una URI y debe indicar el espacio de nombres de atributo en el que el nombre del atributo en cuestión debe ser interpretado. Por tanto, el elemento *Attribute* utiliza el tipo *AttributeDesignatorType* para nombrar el atributo que representa. Además, el elemento *Attribute* contiene un elemento *AttributeValue* para contener el valor del atributo. El elemento *AttributeValue* es de tipo simple *xsi:type* definido en la especificación W3C XML Schema. En el caso en que el valor del atributo fuera una estructura de datos, debería definirse su esquema.

Terminado el estudio del tipo de declaración que permite relacionar un sujeto con unos atributos, pasaremos a estudiar el tipo de sentencia que permite a una autoridad SAML de tipo Punto de Decisión de

Autorización declarar con una decisión de autorización que permita o deniegue a un sujeto el acceso o no a ciertos recursos. Tal y como ya se ha mencionado con anterioridad, el elemento que representa un declaración de decisión de autorización es *AuthorizationStatement*.

6.2.7 Afirmación de autorización

Este tipo de declaración contiene la información generada por una autoridad SAML que confirma que una petición hecha por el sujeto de la declaración para acceder a cierto recurso ha sido aceptada o denegada (decisión de autorización) basándose en algunas evidencias especificadas opcionalmente.

El recurso es identificado mediante una URI. Con el fin de que todas las autoridades y servicios SAML puedan tomar decisiones coherentes deberán ponerse de acuerdo en la forma de interpretar las URIs. La especificación SAML señala que la interpretación de las URIs debe basarse en las reglas definidas en el IETF RFC 2396. Para evitar la ambigüedad en la interpretación, los sistemas SAML deberán intentar siempre manejar la forma normalizada de las URIs. Es decir, siempre que sea posible una autoridad SAML deberá codificar todos los recursos a los que haga referencia mediante la forma normalizada de la URI y, de la misma manera, todas aquellas entidades potenciales de procesar estas URIs que referencian los recursos deberán, antes de poder concluir si dos URIs son equivalentes, formatearlas en su forma normalizada. Pero no sólo vale con normalizar las URIs para evitar una mala interpretación de los recursos referenciados por las mismas. Otro posible punto de

inconsistencia podría surgir cuando existe una mala interpretación entre el recurso que referencia la URI y el sistema de ficheros subyacente. Las partes de una URI son sensibles a las mayúsculas de forma que si el sistema de ficheros subyacentes es sensible a las mayúsculas, un solicitante no debería conseguir el acceso a un fichero tan sólo cambiando de minúsculas a mayúsculas el nombre del mismo contenido en la URI. Por ejemplo, supongamos una URI que hace referencia a cierto recurso hospedado en un sistema de ficheros Windows:

http://www.foo.com/foo/Foo.htm

En este caso, si el sistema de ficheros tuviera el fichero *foo.htm* no se debería permitir el acceso a este recurso puesto que el recurso es Foo.htm y no foo.htm, independientemente de que Windows no distinga entre mayúsculas y minúsculas. En conclusión, no es lo mismo el recurso referenciado por la URI:

http://www.foo.com/foo/Foo.htm

Que el recurso referenciado por:

http://www.foo.com/foo/foo.htm

La sentencia *AuthorizationDecisionStatement* es de tipo *AuthorizationDecisionStatementType* que, a su vez hereda del tipo *SubjectStatementStatementType* de forma que una declaración de este

tipo tiene también asociado un sujeto sobre el que aplica decisión de autorización tomada.

Además este tipo define dos atributos. El primero es obligatorio, se denomina *Resource* y debe ser una URI que referencia el recurso al que el sujeto quiere acceder ejecutando una acción. El segundo de los atributos, también obligatorio, se denomina *Decision* y representa la decisión de la autorización. El tipo de esta atributo es *DecisionType* que es una enumeración de las siguientes cadenas de texto: *Permit*, *Deny*, *o Indeterminate.* Además, la declaración *AuthorizationDecisionStatement* debe reflejar una o varios elementos *Action* que reflejan las acciones que se desean realizar sobre el recurso y puede contener opcionalmente uno o varios elementos *Evidence* que le permita referenciar evidencias sobre las que se ha basado la autoridad SAML para tomar la decisión de autorización.

Un elemento *Action* es una cadena de caracteres que indica la acción más, un atributo opcional, *Namespace*, que es una referencia URI que indica el espacio de nombres de acciones bajo el que se debe interpretar la acción solicitada. Si éste último atributo está ausente se utilizará por defecto la URI predefinida por la especificación cuyo valor es *urn:oasis:names:tc:SAML:1.0:action:rwedc-negation*. Este espacio de nombres se define en la especificación SAML y representa el mismo espacio de nombres que *urn:oasis:names:tc:SAML:1.0:action:rwedc*, pero además permite la negación de las operaciones. Es decir, en el espacio de nombres *urn:oasis:names:tc:SAML:1.0:action:rwedc* se

definen las acciones: Lectura (Read), Escritura (Write), Ejecución (Execute), Borrar (Delete), Control (permite al sujeto definir la política de control de acceso sobre el recurso) y en el espacio de nombres *urn:oasis:names:tc:SAML:1.0:action:rwedc-negation* se definen las mismas pero se admite su negación. Por lo tanto, un sujeto puede solicitar la operación negada de escritura sobre un recurso, y, en el caso en que se le autorice, indicará que efectivamente él no tiene derechos de escritura sobre el recurso. El otro elemento hijo del elemento *AuthorizationDecisionStatement* es *Evidence*, no tiene atributos, y puede contener uno o varios elementos *AssertionIDReference* o *Assertion* que representen las aseveraciones en las que se ha basado la autoridad SAML para tomar la decisión.

Hasta aquí hemos hecho un estudio del conjunto de posibles tipos de declaraciones que los servicios SAML pueden manejar. A continuación estudiaremos el protocolo petición/respuesta que define la especificación y que permite, respectivamente, la solicitud de afirmaciones a una autoridad SAML y su emisión por parte de las mismas.

6.2.8 Protocolo Petición/Respuesta

Las afirmaciones estudiadas pueden ser intercambiadas entre solicitantes y autoridades SAML a través de múltiples protocolos subyacentes. Dentro del conjunto de documentos que se engloban dentro de la especificación SAML se encuentra uno, denominado 'SAML Binding', que define los mecanismos para poder transportar las afirmaciones SAML sobre distintos protocolos. Con el fin de que un solicitante SAML

obtenga alguna de las afirmaciones estudiadas, deberá construir un mensaje de petición, según lo expresa la especificación SAML, en el que puede indicar toda la información necesaria para conseguir la aseveración. Por su parte, la autoridad SAML, potencial emisora de aseveraciones SAML, deberá construir mensajes de respuesta conformes a la especificación en los que pueda devolver toda la información necesaria resultado de su proceso. Es en esta respuesta en la que la autoridad SAML podrá devolver las aseveraciones SAML (elementos *Assertion*) que crea conveniente.

En la siguiente figura se muestra un sencillo esquema del protocolo en cuestión:

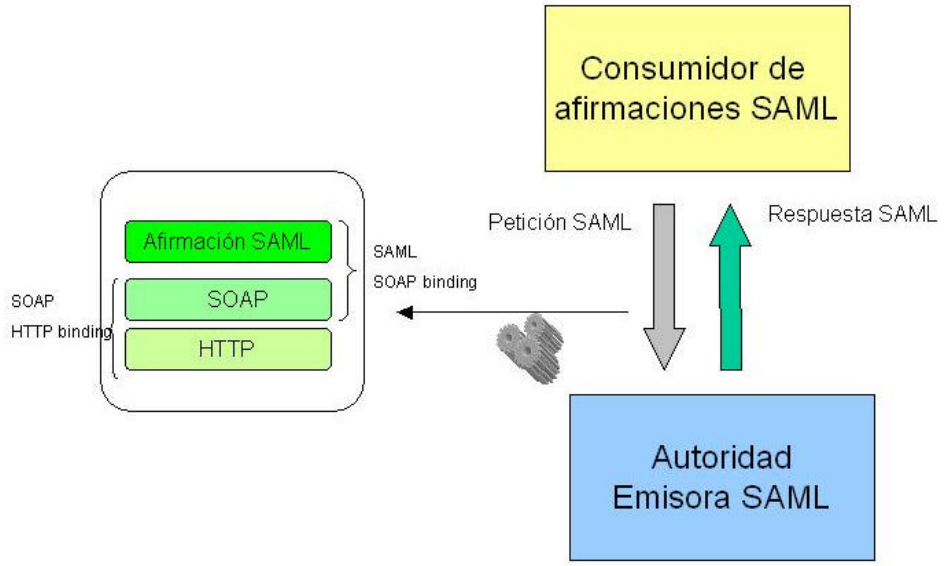

Ilustración 15. Protocolo petición/respuesta SAML.

Por tanto debemos estudiar la estructura de las peticiones y las respuestas SAML con el fin de conocer qué posibilidades y capacidades estándar ya están disponibles. En primer lugar estudiaremos la estructura de una petición SAML.

6.2.9 Peticiones SAML

Una petición SAML está definida mediante el siguiente fragmento de esquema XML:

```
<element name="Request" type="samlp:RequestType" />
    <complexType name="RequestType">
        <complexContent>
            <extension base="samlp:RequestAbstractType">
                <choice>
                    <element ref="samlp:Query" />
                    <element ref="samlp:SubjectQuery" />
                    <element ref="samlp:AuthenticationQuery" />
                    <element ref="samlp:AttributeQuery" />
                    <element
                  ref="samlp:AuthorizationDecisionQuery" />
                    <element ref="saml:AssertionIDReference"
                  maxOccurs="unbounded" />
                        <element
                      ref="samlp:AssertionArtifact"
                      maxOccurs="unbounded" />
                </choice>
            </extension>
        </complexContent>
    </complexType>
```

Como se puede observar el tipo *RequestType* es el que define el elemento *Request*. El elemento *RequestType* hereda de un tipo de elemento abstracto *RequestAbstractType* que define los atributos obligatorios *MinorVersion, MajorVersion, IssueInstant y RequestID* . Además, puede opcionalmente contener un elemento *ds:Signature* y contener cero o varios elementos *RespondWith* que representan el tipo de afirmación solicitada por el solicitante y cuyo tipo es *QName* tal y como se espefica en la especificación *W3C XML Schema*. Por ejemplo, si un solicitante desea que su petición sea respondida con una aseveración que contenga una declaración de autorización construirá un elemento *RespondWith* de la siguiente manera:

<RespondWith>**saml:AuthorizationStatement**</RespondWith>

NOTA: Tal y como se expresa en la especificación el uso del elemento RespondWith está obsoleto y, siempre que se pueda, no debe ser utilizado:

> *"NOTE: This element is deprecated; use of this element SHOULD be avoided because it is planned to be removed in the next major version of SAML. "*

Descrito el tipo abstracto *RequestAbstractType* pasemos a analizar el tipo *RequestType*. Como ya se ha dicho *RequestType* hereda los atributos y elementos definidos en el tipo *RequestAbstractType* y además puede elegir entre alguno de los siguientes elementos como elemento hijo: *AssertionIDReference*, *AssertionArtifact*, *Query*, *SubjectQuery*, *AuthenticationQuery*, *AttributeQuery*, *AuthorizationQuery*. Analicemos uno a uno cada unos de estos elementos.

Si el elemento es *AssertionIDReference* estará solicitando el elemento *Assertion* cuyo identificador se corresponda con el indicado en el elemento.

Si el elemento *Request* contiene un elemento *AssertionArtifact* entonces estará solicitando el elemento *Assertion* que representa el artefacto SAML. Veremos qué es un artefacto SAML y cómo se utiliza cuando estudiemos los perfiles definidos por SAML.

El elemento *Query* representa una petición genérica y sirve como punto de extensión para crear peticiones personalizadas y poder así aprovechar el marco de trabajo petición/respuesta SAML con otro tipo de afirmaciones. Si además sabemos que las peticiones personalizadas siempre requerirán un sujeto (el que realiza la petición) se puede utilizar el elemento *SubjectQuery*. El elemento *SubjectQuery* se define como un elemento *SubjectQueryAbstractType* que contiene un elemento *Subject*

Si el elemento *Request* escoge como elemento hijo un elemento *AuthenticationQuery*, *AuthorizationQuery* o *AttributeQuery* entonces estará solicitando una afirmación como respuesta que incluya una declaración *AuthenticationStatement*, *AuthorizationStatement* respectivamente o *AttributeStatement*.

El elemento *AuthenticationQuery* es de tipo *AuthenticationQueryType* que hereda del tipo de definición *SubjectQueryAbstractType* (lo que significa que refleja un sujeto mediante el elemento *Subject*) y agrega tan sólo la definición del atributo opcional *AuthenticationMethod* que sirve como filtro para las afirmaciones de autenticación solicitadas. Por ejemplo, en el caso en que no se especifique este atributo podríamos construir una petición que solicitara una afirmación de autenticación para el sujeto indicado en el elemento *Subject*. Sin embargo, indicando este atributo es posible crear peticiones cómo: “Dame las afirmaciones SAML en las cuales el sujeto indicado en el elemento *Subject* se haya autenticado mediante el mecanismo indicado en el atributo *AuthenticationMethod*”. Como a estas alturas ya es imaginable, el tipo

del atributo *AuthenticationMethod* es una referencia URI que indica el algoritmo de autenticación requerido. En las posibles afirmaciones de respuesta se deberá reflejar el sujeto enviado en la petición. Además, si el atributo *AuthenticationMethod* estuvo presente en la petición, en el conjunto de afirmaciones de respuesta debe existir al menos una que contenga una declaración de autenticación que haya sido completada mediante ese algoritmo de autenticación.

El elemento que solicita afirmaciones que contengan declaraciones de decisiones de autorización es *AuthorizationDecisionQuery*. El tipo que define este elemento, *AuthorizationDecisionQueryType*, hereda del tipo *SubjectQueryAbstractType* y, por tanto, siempre incluirá un sujeto en las peticiones (sujeto con el que se vincula la decisión final de la autorización). Cuando un solicitante SAML envía una petición *Request* con este elemento puede realizar preguntas cómo: "¿Podrían ejecutarse estas acciones sobre este recurso dado el sujeto y esta evidencia?".

El tipo de este elemento se especifica a continuación:

```
<element name="AuthorizationDecisionQuery"
        type="samlp:AuthorizationDecisionQueryType" />
        <complexType name="AuthorizationDecisionQueryType">
            <complexContent>
                <extension
            base="samlp:SubjectQueryAbstractType">
                    <sequence>
                      <element ref="saml:Action"
                maxOccurs="unbounded" />
                      <element ref="saml:Evidence"
                minOccurs="0" />
```

```
                    </sequence>
                <attribute name="Resource" type="anyURI"
            use="required" />
                    </extension>
                </complexContent>
        </complexType>
```

Según se expresa en el fragmento anterior que define el tipo del elemento *AuthorizationDecisionQueryType* vemos como se componen de un atributo requerido *Resource*, y de dos elementos (más los atributos y elementos heredados el tipo *SubjectQueryAbstractType) Action* y *Evidence*. El atributo *Resource* es, acorde con lo que ya hemos estudiado, el recurso para el cual el solicitante SAML está solicitando acceso y se indica mediante una URI. Por su parte, el elemento *Action* señala la acción para la que se está solicitando el acceso (p.e: *urn:oasis:names:tc:SAML:1.0:action:r* para lectura*)* y el elemento *Evidence* contiene una evidencia que puede ser utilizada por el Punto de Decisión de Autorización SAML en su toma de decisión de permisibilidad del acceso.

Otro tipo de petición más es aquella que permite obtener el valor de ciertos atributos de cierto sujeto. La respuesta esperada a este tipo de petición sería una afirmación de atributos cuya estructura ya hemos analizado. El elemento que refleja este tipo de afirmaciones es *AttributeQuery* y su sintáxis es muy sencilla. Su tipo de elemento hereda del tipo abstracto *SubjectQueryAbstract*, por lo que toda petición reflejará un sujeto, aquel de quién se estará solicitando el valor de los

atributos. Además define un atributo y un elemento. El atributo, cuya inclusión es opcional, es *Resource* y sirve para referenciar un recurso para el cual se está solicitando el valor de uno o varios de sus atributos. Su valor práctico es evaluar si el sujeto de la petición tiene o no acceso al valor del atributo del recurso. En cuanto al elemento (que puede ocurrir ninguna o muchas veces), es *AttributeDesignator* que, recordemos por el estudio hecho con las aseveraciones de atributo, sirve para nombrar un atributo (aquel del que queremos conocer su valor).

Con la petición de atributos ya hemos estudiado todos los posibles tipos de peticiones que un cliente SAML puede realizar a una autoridad SAML. A continuación proseguiremos el estudio analizando el formato de las posibles respuestas que pueden darse para los tipos de peticiones estudiados.

6.2.10 Respuestas SAML

Al igual que ocurría con el tipo de dato abstracto *QueryAbstractType* para las peticiones, el tipo de datos *ResponseAbstractType* es el tipo de datos base para todos los posibles tipos de respuestas generables. Este tipo abstracto incluye:

- Un atributo obligatorio *ResponseID* de tipo *xsd:ID* que identifica unívocamente la respuesta.
- Un atributo opcional *InResponseTo* que es un identificador al atributo *RequestID* de la petición a la que se asocia la respuesta.
- Los ya mencionados atributos obligatorios *MinorVersion* y *MajorVersion* para dar una versión en la respuesta.

- El atributo requerido *IssueInstant* que refleja en UTC el instante en el tiempo en el que se generó la respuesta.
- Un atributo opcional *Recipient* que indica el receptor de esta respuesta. Este atributo es muy útil contra ataques de re-envío de los mensajes hacia destinatarios no autorizados, una protección establecida como requisito indispensable en muchos perfiles de uso de SAML. Su valor es una URI que referencia al receptor.
- Un elemento opcional *ds:Signature* que permite incluir una firma sobre los datos contenidos en la respuesta.

El elemento *Response* es de tipo *ResponseType* que, como podemos imaginar extiende del tipo *ResponseAbstractType* y sirve para reflejar el estado de la petición SAML correspondiente a una petición SAML además de una lista de cero o más afirmaciones que dan respuesta a la solicitud. El tipo *ResponseType* agrega a la definición del tipo *ResponseAbstractType* la definición de dos elementos: *Status* (obligatorio) y *Assertion* (puede aparecer cero o varias veces).

El primer elemento, *Status*, refleja el estado fruto de la ejecución de la petición por parte de la autoridad SAML. Este elemento contiene un elemento obligatorio, *StatusCode*, que indica el código que representa el estado de la correspondiente petición, y dos elementos opcionales, *StatusMessage* y *StatusDetail*. El primero es un mensaje descriptivo resumen del estado de la respuesta y el segundo es información detallada que complementa la información aportada por el elemento anterior.

El elemento *StatusCode* define a su vez un atributo obligatorio *Value* que es tipo *QName* y puede incluir elementos *StatusCode* anidados. Los valores posibles pre-definidos del tributo *Value* están recogidos en la especificación SAML e incluye aquellos que permiten expresar que el procesamiento de la petición fue llevado a cabo con éxito, que se encontró un error por no coincidir las versiones (*MinorVersion* y *MajorVersion*) reflejadas en la petición, etc.

La especificación SAML permite que las entidades SAML definan más códigos de error siempre que sean definidos en un espacio de nombres distinto al de la especificación SAML.

Por último, y para cerrar esta sección de respuestas SAML, debemos mencionar como en el apartado ***3.4.4 Response to Queries*** de la especificación se señala que en toda respuesta a una petición SAML, cada afirmación devuelta por la autoridad SAML debe existir al menos una sentencia cuyo elemento *Subject* coincida con el elemento *Subject* encontrado en la petición. Dos elementos *Subject* coinciden en el caso de que si uno incluye el elemento *NameIdentifier* entonces el otro incluye uno idéntico y, si uno incluye un elemento *SubjectConfirmation*, entonces el otro incluye otro idéntico.

Con lo estudiado hasta ahora hemos recorrido los 3 primeros capítulos de la especificación fundamental de SAML. El resto de la especificación cuenta el sistema de versiones de la especificación SAML, cómo combinar la especificación de firmas digitales (XML Signature Syntax

and Processing) con la de SAML, las extensiones SAML y los identificadores preestablecidos por la especificación para distintos tipos de elementos que maneja.

Como estamos analizando las especificaciones bajo el prisma de la seguridad, aquí tan sólo analizaremos el contenido del capítulo 5 que explica como utilizar correctamente las firmas digitales en los elementos definidos por la especificación SAML.

6.2.11 Integración de SAML con XML Digital Signature

Como ya hemos señalado en varios ocasiones a lo largo de las secciones anteriores SAML hace uso del mecanismo de firma digital en diversas ocasiones: para firmar las afirmaciones, y para firmar las peticiones/respuestas SAML.

Las firmas digitales que firman una afirmación emitida por una autoridad SAML proporcionan:

- Integridad del contenido.
- Autenticación de la autoridad SAML que emitió la afirmación.
- Si la firma está realizada mediante la clave privada de la autoridad SAML además proporciona no repudio del origen.

En el caso de las firmas sobre las peticiones/respuestas SAML:

- Se garantiza la integridad de los mensajes.
- Se autentica frente al receptor la parte emisora del mensaje.
- Si la firma está basada en la clave privada del emisor del mensaje, entonces también se proporciona no repudio del origen.

Las firmas digitales no son de uso obligatorio según la especificación SAML. Por ejemplo:

- En algunas situaciones una afirmación no firmada puede heredar una firma aplicada sobre entidades que la contienen. Por ejemplo, una aseveración no firmada puede ganar esta propiedad si se introduce dentro de un mensaje de respuesta del protocolo SAML firmado. Las firmas digitales "heredadas" deberían ser utilizadas con cuidado sobre todo cuando el objeto contenido (como por ejemplo una aseveración) posee un tiempo de vida persistente y no transitorio. El motivo de esta advertencia es que, para que la firma siempre sea válida, se debe retener todo el contexto para conseguir que su validación siempre sea correcta.
- Las firmas digitales tampoco son de uso obligatorio cuando los canales de comunicaciones entre las partes SAML son los suficientemente seguros.

Salvo en estas dos situaciones la especificación recomienda el uso de las firmas digitales, específicamente en los siguientes casos:

- Una afirmación SAML debe firmarse cuando es recibida por un cliente SAML procedente de una entidad que no es una autoridad SAML.
- Un mensaje del protocolo SAML debe ser firmado cuando se llega a un destinatario procedente desde una entidad que no es la parte SAML que generó el mensaje original.

6.2.12 Vínculos SAML

Hasta ahora en este capítulo de SAML hemos realizado, en primer lugar, un estudio por las estructuras de datos XML que mantienen los distintos tipos de afirmaciones SAML y, en segundo lugar, hemos analizado el formato de los mensajes del protocolo petición/respuesta que define la especificación. En este apartado vamos a realizar un breve resumen del documento de la especificación SAML, titulado *Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) V1.1*, que describe los mecanismos necesarios para transportar estos mensajes de nivel SAML sobre protocolos de transporte de más bajo nivel (SOAP).

El mapeo entre el protocolo petición/respuesta definido por SAML con un protocolo de mensajes o de comunicación estándar se denomina vínculo del protocolo SAML *(SAML protocol binding)*. Recordemos que este término surge en otras especificaciones como por ejemplo en la especificación SOAP cuando define como mapear los mensajes SOAP en protocolos de transporte de más bajo nivel como HTTP o STMP.

Un *FOO binding*, en términos de SAML, sería la definición de una correlación del protocolo de petición/respuesta con el protocolo *FOO*. Por ejemplo, *SOAP binding* describe la forma en la que el intercambio de las peticiones y respuestas SAML se mapean en intercambios de mensajes SOAP.

Otro término acuñado por la especificación, y que guarda relación con el término *binding*, es *profile* (perfil en español). Un *profile o* perfil SAML define el conjunto de reglas que especifican la forma en la que las afirmaciones SAML pueden ser insertadas y extraídas dentro de mensajes de cierto protocolo o marco de trabajo. Un ejemplo de perfil SAML podría ser el perfil SOAP que especifica cómo las afirmaciones SAML pueden ser insertadas dentro de mensajes SOAP, cómo las cabeceras SOAP se ven afectadas por las afirmaciones SAML y cómo se deberían reflejar errores relativos a SAML en el mensaje SOAP.

Este documento de la especificación SAML define un conjunto determinado de *bindings* y *profiles* con ciertos protocolos. El comité técnico encargado de la especificación SAML (*OASIS Security Services Technical Committee 184*) aclara en dicho documento que por razones obvias de límite en sus recursos le es imposible encargarse de todas las estandarizaciones de todas las posibles tecnologías que surjan en un futuro y ofrece unas directrices básicas a seguir a la hora de definir un nuevo *binding* o un *profile* así como el proceso a seguir para llegar a registrarlo como estándar (apartado *2.2 Process Framework for Describing and Registering Protocol Bindings and Profiles*).

La especificación sólo tiene estandarizado el vínculo SOAP. Si recordamos del estudio realizado sobre el protocolo SOAP, en cualquier mensaje SOAP se incluía en una “envoltura” SOAP la cual incluía dos partes claramente diferenciadas, la cabecera y el cuerpo. En la cabecera del mensaje SOAP se incluyen bloques de cabecera cuyo objetivo es

proporcionar metainformación sobre los datos, denominado *payload* en inglés, incluidos en el cuerpo SOAP del mensaje. Los elementos del protocolo petición/respuesta SAML serán incluidos dentro de cuerpo del mensaje SOAP. Básicamente, el sistema utilizado para realizar conversaciones SAML sobre SOAP se compone de un mensaje SOAP, que representa la petición, en cuyo cuerpo se incluye un elemento *Request* del espacio de nombres SAML, y de un mensaje SOAP de respuesta en cuyo cuerpo se incluye un elemento *Response* SAML o código de fallo SOAP. El solicitante SAML puede incluir cualquier tipo de información dentro de la cabecera SOAP. La autenticación del solicitante y del servicio SAML y la integridad y confidencialidad de los mensajes son aspectos que la especificación determina como opcionales y para los que no define mecanismos específicos.

Al igual que SAML tiene un vínculo con SOAP, SOAP lo puede tener con protocolos de más bajo nivel como HTTP o SMTP. En este caso, la especificación determina como obligatorio que sea el vínculo SOAP sobre HTTP el que sea utilizado cuando se utiliza el vínculo SOAP de SAML. Así cualquier procesador SAML que se declara conforme a la especificación *SAML SOAP binding* debe implementar SAML sobre SOAP sobre HTTP. El SOAP *binding* sobre HTTP requiere el uso de la cabecera *SOAPAction* como parte de una petición HTTP SOAP y, por tanto, un solicitante SAML podría establecer el valor de dicha cabecera como:

http://www.oasis-open.org/committees/security

El siguiente mensaje representa una petición SAML de autenticación que utiliza el *SOAP binding*:

```
POST /SamlService HTTP/1.1
Host: www.example.com
Content-Type: text/xml
Content-Length: nnn
SOAPAction: http://www.oasis-open.org/committees/security
<SOAP-ENV:Envelope
xmlns:SOAPENV="http://schemas.xmlsoap.org/soap/envelope">
<SOAP-ENV:Body>
<samlp:Request xmlns:samlp:="..." xmlns:saml="..." xmlns:ds="...">
<ds:Signature> ... </ds:Signature>
<samlp:AuthenticationQuery>
...
</samlp:AuthenticationQuery>
</samlp:Request>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Su correspondiente respuesta podría ser:

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: nnnn

<SOAP-ENV:Envelope xmlns:SOAP-
      ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Body>
      <samlp:Response xmlns:samlp="...." xmlns:saml="..."
            xmlns:ds="..">
         <Status>
            <StatusCode Value="samlp:Success" />
         </Status>
         <ds:Signature>...</ds:Signature>
         <saml:Assertion>
            <saml:AuthenticationStatement>...</saml:Authe
                  nticationStatement>
```

```
            </saml:Assertion>
        </samlp:Response>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

En cuanto a los perfiles, que recordemos son el conjunto de reglas que determinan cómo introducir y extraer afirmaciones SAML de mensajes de otros protocolos o marcos de trabajo, se definen dos:

- *Web Browser SSO Profile of SAML.*
- *Browser/POST Profile of SAML.*

Ambos perfiles se basan en un navegador Web como cliente SAML y se definen con el objetivo de que soporten *Single Sign-On*. Para cada perfil, la especificación define un modelo de posibles amenazas así como las posibles contramedidas que pueden ser adoptadas. Es de destacar que ya existen algunos perfiles publicados, de manera independiente al comité *OASIS Security Services Technical Comitte,* por organizaciones como el *Liberty Alliance Project* que ha publicado diversos perfiles de su versión extendida de SAML, u otro comité del propio OASIS como el *OASIS Servicio Webs Security Technical Comitte* que ha publicado, todavía en borrador, un perfil denominado *SAML token profile* que define cómo utilizar las afirmaciones SAML en el bloque de cabecera *wsse:Security* para securizar los mensajes definidos por la especificación WS-Security.

6.2.13 Perfil SAML Web Browser SSO

Los escenarios de este perfil son aquellos en los que un cliente, a través de un navegador Web, se autentica en un sitio Web origen (denominado *sitio de origen* por la especificación) y, a partir de este momento, podrá

acceder a otros sitios Web destino sin necesidad de autenticarse y utilizando un recurso seguro (p.e: una cookie).

Los perfiles de este tipo deben asumir que el usuario se ha autenticado en un sitio Web origen mediante algún mecanismo fuera del ámbito de SAML y utilizando un navegador Web estándar. Además, también deben asumir que el sitio Web origen es capaz de mantener una traza de los usuarios que se han autenticado localmente. Normalmente esto se materializa en el mantenimiento de sesiones que podrían estar representadas por una cookie o una URL codificada. Más tarde, el usuario intenta acceder a un recurso objetivo en el sitio Web de destino y, tras uno o más pasos posteriores (redirecciones), el usuario llega hasta un *servicio de transferencia entre sitios Web* ubicado el sitio Web de origen. A partir de este punto el perfil en cuestión describe una serie de intercambios de mensajes HTTP que acaban transfiriendo el navegador del usuario a un *servicio procesador de afirmaciones* que estará ubicado en el sitio Web de destino. La información sobre las afirmaciones SAML proporcionadas por el sitio Web de origen y que están asociadas con el usuario solicitante y el recurso solicitado es transmitida del sitio Web origen al sitio Web destino mediante el protocolo de intercambio mencionado.

Ahora, el sitio Web de destino a partir de la información incluida en las afirmaciones SAML y la información del recurso al que se desea acceder tomará la decisión de si autorizar o no el acceso. De esta forma el cliente sólo habrá tenido que autenticarse una vez y sin embargo habrá podido

acceder al recurso de un sitio Web distinto a aquel con el que se autenticó. Este proceso se denomina *Single Sign-On (SSO).* Obviamente, entre el sitio Web origen y el sitio Web destino deberá existir una relación de confianza. La siguiente figura ilustra la plantilla para conseguir SSO.

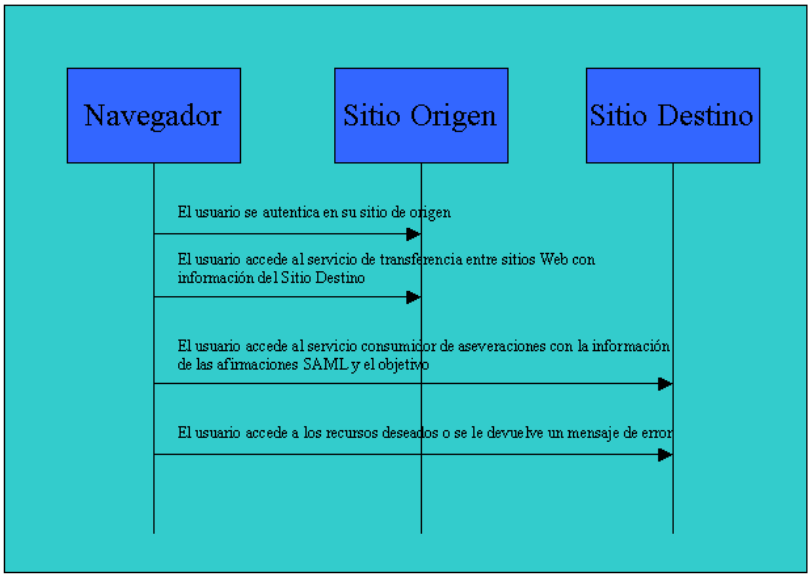

Ilustración 16. Visión genérica del protocolo definido por el perfil SAML Web Browser SSO.

Existen dos técnicas en el perfil Web Browser SSO que especifican cómo transmitir la información de un sitio a otro vía un navegador Web estándar.

- Artefacto SAML.

- Formulario POST.

El perfil artefacto/navegador de SAML se basa en una referencia a la afirmación necesaria que viaja en un artefacto SAML, la cual el sitio Web de destino de-referencia para obtener desde el sitio Web origen la afirmación SAML de autenticación y así poder determinar si el cliente está o no autenticado. Algo que es importante y se debe destacar es el motivo de utilizar un artefacto "pequeño". Esto es debido a la restricción en el tamaño de las URLs impuesta por casi todos los navegadores y servidores Web (normalmente establecida a 2000 caracteres).

El perfil artefacto/navegador necesita de tan sólo una interacción entre las tres partes implicadas (el cliente equipado con navegador Web, el sitio Web de origen y el sitio Web de destino), junto con una sub-interacción entre dos de ellas (el sitio Web origen y el sitio Web destino). La secuencia de interacciones que se lleva a cabo se muestra en la siguiente figura:

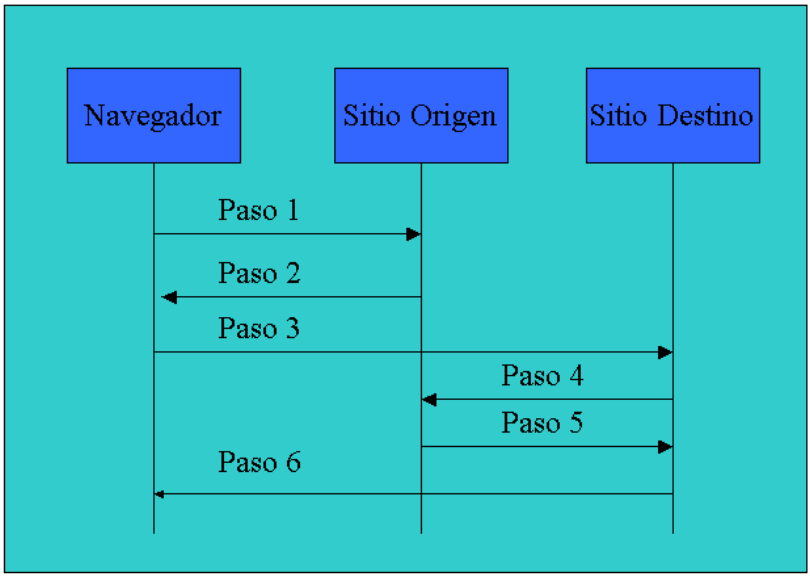

Ilustración 17. Pasos definidos por el perfil Web Browser SSO.

Este perfil hace uso de la URL y por tanto utiliza la terminología definida en el RFC 1738 que describe los componentes de una URL. Una URL HTTP tiene la siguiente forma:

http://<HOST>:<port>/<path>?<searchpart>

En el paso 1, el navegador del usuario accede al servicio de transferencia entre sitios Web ubicado en el host *https://<servicio-de-transferencia entre-sitios-Web>*, con información sobre el recurso deseado y ubicado en el sitio Web de destino reflejado en la URL.

En el paso 2, el *servicio de transferencia entre sitios Web* responde y redirige el navegador del usuario al *servicio consumidor de aseveraciones* localizado en el sitio Web de destino. La URL utilizada por el sitio Web de origen para redirigir al cliente al servicio consumidor de aseveraciones del sitio Web destino se denomina *URL del receptor del artefacto.*

La respuesta HTTP podría tener la siguiente forma:

<HTTP-Version> 302 <Reason Phrase>
<other headers>
Location : https://<camino y nombre de host del receptor del artefacto>?*<SAML searchpart>*
<other HTTP 1.0 or 1.1 components>

Donde *<camino y nombre de host del receptor del artefacto>* proporcionan el nombre del host, el puerto y los componentes del camino del receptor del artefacto asociado con el servicio consumidor de aseveraciones localizado en el sitio de destino. Por su parte el fragmento *<SAML searchpart>* se descompone en los siguientes elementos:

…TARGET=<Recurso objetivo>…SAMLart=<artefacto SAML> …

Se debe incluir una única descripción del objetivo y al menos un artefacto SAML aunque podrían ser incluidos varios. Mediante esta parte

el cliente puede señalar al servicio consumidor de aseveraciones el recurso al que quiere acceder y las afirmaciones que le autentican.
De acuerdo a HTTP 1.1y HTTP 1.0, se recomienda el uso del código de error 302 para indicar que “el recurso solicitado reside temporalmente en una URI diferente”.

Debería garantizarse tanto la confidencialidad como la integridad del mensaje del paso 2. Por eso, la especificación recomienda que el servicio de transferencia entre sitios Web utilice SSL 3.0 o TLS 1.0. De otra forma, los artefactos SAML podrían ser interceptados por atacantes pudiendo suplantar la identidad del sujeto.

En el paso 3, el navegador del usuario accede al servicio receptor del artefacto ubicado en el host *https://<nombre-de-host-del-receptor-del artefacto>* que incluye el artefacto SAML que representa la información de autenticación del usuario asociada a la URL. La petición HTTP contendrá una parte en la que se especificará la información relacionada con SAML y en la que se incluirá un único recurso objetivo y uno o varios artefactos SAML.

Al igual que en el paso anterior, se debe garantizar la confidencialidad y la integridad del mensaje del paso 3.
En los pasos 4 y 5 se adquieren las afirmaciones SAML reales. El sitio Web destino, deshace la referencia a todos los artefactos SAML recibidos con el fin de obtener las afirmaciones SAML correspondientes a cada uno de ellos. Estos pasos utilizarán el protocolo de intercambio de

mensajes SOAP de forma que se deberá escoger un vínculo SOAP con un protocolo subyacente que, como veremos más adelante deberá garantizar ciertos requisitos de seguridad, y que permita el intercambio de las peticiones de las afirmaciones SAML además de sus correspondientes respuestas entre el sitio Web origen y el sitio Web destino. Recordando el protocolo de petición/respuesta descrito en secciones anteriores, se utilizarán peticiones que incluirán los artefactos SAML (peticiones SAML que incluyen elementos *AssertionArtifact*) en lugar de peticiones de tipos de aseveraciones específicas (*AuthorizationQuery*, *AuthenticationQuery*, *etc.*).

El sitio Web destino actuará como solicitante SAML y, por su parte, el sitio Web origen como un servicio SAML de afirmaciones. El sitio Web destino debe enviar un mensaje *samlp:Request* al sitio Web origen solicitando las afirmaciones mediante el envío de los artefactos SAML recibidos. Si el sitio Web origen es capaz de encontrar y construir las afirmaciones solicitadas, responde con un mensaje *samlp:Response* que incluye las afirmaciones solicitadas. De otra forma, contestará con un mensaje *samlp:Response* sin aseveraciones. El elemento *samlp:Status* del mensaje de respuesta *saml:Response* debe incluir un elemento *samlp:StatusCode* con valor *Success.* En el caso de que el sitio Web origen devuelva las aseveraciones deberá devolver exactamente el mismo número de aseveraciones que las solicitadas. En el caso en que no fuera así, el sitio Web destino deberá tratar el mensaje como incorrecto.

El sitio Web origen debe implementar un mecanismo que le permita devolver cierta afirmación sólo una vez. Es decir, la misma afirmación SAML, una vez devuelta en una interacción petición/respuesta como la descrita deberá quedar invalidada para ser devuelta en posteriores interacciones. Algunas implementaciones solucionan este punto eliminando del almacenamiento persistente la aseveración la primera vez que es devuelta. Si llegará una segunda petición que solicitará la misma aseveración ya enviada, el sitio Web origen deberá tratar la petición de la misma manera que si hubiera recibido una petición de un artefacto SAML desconocido. El vínculo con el protocolo SAML escogido para la interacción deberá proporcionar confidencialidad e integridad de los mensajes así como autenticación de las dos partes.

El sitio Web origen debe devolver una respuesta SAML sin aseveraciones en aquellos casos en los que reciba un mensaje de petición *samlp:Request* desde un sitio Web destino X que contenga un artefacto emitido por el sitio Web origen para otro sitio Web destino Y. Suponiendo, claro está, que los sitio Web destino X e Y son distintos.

Al menos una de las aseveraciones SAML devueltas al sitio Web de destino debe ser un SAML SSO. Las declaraciones de autenticación podrían incluirse en más de una de las afirmaciones devueltas. Cada declaración basada en un sujeto e incluida en la(s) aseveración(es) devueltas al sitio Web de destino deben contener un elemento *saml:SubjectConfirmation* con la siguiente información:

- El elemento *saml:ConfirmationMethod* debe tener asignado el valor *urn:oasis:names:tc:SAML:1.0:cm:artifact*.

- El elemento *saml:SujectConfirmationData* no debe especificarse. Recordemos que este elemento puede contener información adicional que podría ser consultada con el objetivo de verificar la identidad del sujeto de la aseveración.

En función a la información obtenida en las afirmaciones recuperadas por el sitio Web destino se podría desencadenar el intercambio de mensajes adicionales con el sitio Web origen.

Finalmente, en el paso 6, el navegador del usuario recibe una respuesta HTTP que permite o deniega el acceso al recurso deseado.

Un aspecto por el que hemos pasado rápidamente es el formato del artefacto SAML. Recordemos que el artefacto SAML es el elemento que el sitio Web origen, mediante su servicio de transferencia entre sitios Web, incluye en la *URL receptor de artefacto* redirigida al servicio consumidor de afirmaciones del sitio Web destino y, a partir del cual, éste último obtiene las aseveraciones SAML oportunas. Bien, pues este artefacto SAML se compone de un código obligatorio de dos bytes que representa el tipo del artefacto:
SAML_artifact := B64(TypeCode RemainingArtifact)
TypeCode := Byte1Byte2

El perfil en cuestión define un código de tipo artefacto con valor 0x0001, que es obligatorio en cualquier implementación del perfil navegador/artefacto. El artefacto se define de la siguiente manera:

TypeCode := 0x0001
RemainingArtifact = SourceID AssertionHandle
SourceID := 20-byte_sequence
AssertionHandle := 20-byte_sequence

El campo *SourceID* es una secuencia de 20 bytes utilizada por el sitio Web destino para determinar la identidad del sitio Web origen y su ubicación. Se asume que el sitio Web destino mantendrá una tabla de valores de *SourceIDs* así como de la URL o dirección del correspondiente servicio SAML al que solicitar las afirmaciones. Esta información es comunicada entre el sitio Web origen y el sitio Web destino por mecanismos fuera de línea. Cuando el sitio Web destino recibe el artefacto SAML, determina si el *SourceID* pertenece a un sitio Web origen conocido y obtiene la ubicación del servicio SAML.

En cuanto a la construcción de los valores del campo *AssertionHandle* se rige por el principio de que no debería tener una relación predecible con los contenidos de la afirmación referenciada en el sitio Web origen y debe ser imposible construir o adivinar el valor de un *AssertionHandle* válido. Para más detalle, la especificación propone una serie de recomendaciones que el sitio Web origen puede adoptar cuando crea los artefactos SAML.

A continuación estudiaremos un modelo de las posibles amenazas relacionadas con el perfil descrito. Además se propondrán medidas de prevención para mitigar los riesgos de seguridad.

La primera amenaza contemplada se denomina Robo del Artefacto SAML (*Stolen artifact*) y supone que un posible atacante se hace con un artefacto válido emitido por un sitio Web origen de forma que podría suplantar la identidad del usuario en el sitio Web destino. Recordemos que el artefacto SAML es generado por el sitio Web origen e incluido en la *URL del receptor del artefacto* que referencia al servicio consumidor de aseveraciones localizado en el sitio Web destino. Si el atacante rastrea (sniffing) la URL (recordemos que realmente el sitio Web origen devuelve una contestación HTTP 302 en la que incluye el/los artefactos/s) y se hace con los artefactos podría dirigirse él mismo al sitio Web destino suplantando la personalidad del usuario real. Las contramedidas son la garantía de la confidencialidad en los pasos 2, 3, 4 y 5. Si aún así, el atacante es capaz de obtener el valor de los artefactos, se podrían establecer contramedidas adicionales como:

- El sitio Web origen y el sitio Web destino deberían realizar un esfuerzo considerable para asegurar que la configuración de sus relojes difiere en unos pocos minutos a lo sumo.
- Las afirmaciones SAML comunicadas en el paso 5, que recordemos es aquel en el que el sitio Web de origen devuelve las aseveraciones SAML a cambio de los artefactos SAML al sitio Web destino, deben incluir una afirmación SSO.

- El sitio Web origen debería mantener una traza de la diferencia de tiempos entre el momento en el que el artefacto SAML es generado y ubicado en la URL y el momento en el que llega la petición *samlp:Request* que transporta el artefacto recibido desde el sitio Web destino. Se recomienda un máximo de unos minutos de forma que si llegan peticiones de afirmaciones en forma de artefactos SAML que han superado el límite el sitio Web origen no se debería contestar con las aseveraciones solicitadas.
- Es posible que el sitio Web origen genere afirmaciones SSO SAML tanto en el caso en el que es creado el correspondiente artefacto SAML como cuando se recibe un mensaje *samlp:Request* que transporta el artefacto SAML. El periodo de validez de la afirmación debe ser establecido de manera apropiada en cada caso: más largo en el primero y más corto en el segundo.
- Los valores de los atributos *NotBefore* y *NotOnOrAfter*, que recordemos pertenecen al elemento *Conditions* contenido en cualquier tipo de aseveración SAML, debe tener un periodo de validez lo más corto posible, normalmente del orden de unos pocos minutos. Esto asegura que si el artefacto es robado puede ser utilizado solamente durante una ventana de tiempo muy pequeña.
- El sitio Web destino debe comprobar el periodo de validez de todas las aseveraciones obtenidas desde el sitio origen y rechazar todas aquellas que hayan expirado. Un sitio Web destino podría implementar una prueba de validación más estricta para las

afirmaciones SSO como por ejemplo requerir atributos *IssueInstant* o *AuthenticationInstant*.

- Si una afirmación de autenticación incluye un elemento *saml:SubjectLocality* con la dirección IP del usuario, el sitio Web destino podría chequear la IP del navegador contra la dirección IP contenida en la declaración de autenticación. Recordemos que el elemento *SubjectLocality* es uno de los dos posibles elementos a incluir dentro de una afirmación *AuthenticationStatement*. El otro es *AuthorityBinding* que referencia a una autoridad SAML que podría proporcionar información adicional sobre la identidad del principal autenticado.

Otro tipo de ataque que podría darse es aquel dirigido al protocolo SAML de petición/respuesta producido en los pasos 4 y 5. El patrón de intercambio de mensajes podría sufrir ataques de robo de afirmaciones o artefactos; de inserciones o modificaciones de mensajes; ataques *"man-in-the-middle"*. El requisito de que el protocolo de vinculación SAML utilizado autentique obligatoriamente a ambas partes, y garantice la integridad y confidencialidad de los mensajes suponen la defensa contra estos tipos de ataques.

Otro posible tipo de ataque es aquel en el que un sitio Web de destino, puesto que recibe los artefactos SAML que prueban la autenticación del usuario, pueda utilizarlos maliciosamente para suplantar al usuario frente a otro sitio Web destino. El nuevo sitio Web destino recibirá los artefactos SAML y obtendrá del sitio Web original las afirmaciones del

usuario real permitiendo que el sitio Web destino atacante le suplante. En este escenario el nuevo sitio Web destino deberá autenticarse frente al sitio Web original para poder obtener las aseveraciones SAML correspondientes a los artefactos SAML. Existen dos posibilidades a considerar:

- Si el nuevo sitio Web destino no tiene relación de confianza con el sitio origen, será imposible que se pueda autenticar frente a él y por tanto el ataque fallará.
- Sin embargo, si el nuevo sitio Web destino posee una relación de confianza con el sitio Web origen, el sitio Web origen determinará si las aseveraciones están siendo solicitadas por el sitio Web destino para quién originalmente se emitieron los artefactos SAML. En tal caso, el sitio Web origen no deberá facilitar las aseveraciones al nuevo sitio Web destino.

Otro tipo de ataque posible más, es aquel que tiene que ver con la creación de artefactos SAML. Un usuario malicioso podría generar artefactos SAML. Como ya hemos estudiado, la especificación proporciona una serie de normas a seguir en la creación de los artefactos SAML de forma que no parece factible adivinar o construir el valor de un *AssertionHandle* real y válido. Un usuario malicioso podría intentar adivinar repetidamente el valor de una artefacto SAML (uno que se corresponde con una afirmación existente en un sito Web origen) aunque dado el tamaño del espacio de posibles valores, esta acción le supondría un número enorme de intentos fallidos. Un sitio Web origen debería

implementar una serie de medidas para asegurar que el intento repetido de peticiones utilizando un artefacto inexistente resulten en una alarma.

El último ataque reflejado por la especificación para el perfil SAML navegador/artefacto SSO es aquel inherente a la exposición del estado del navegador Web. Este perfil implica que el navegador se descargue los artefactos SAML al navegador Web desde un sitio Web origen. Esta información se encontrará disponible como parte del estado del navegador Web y, normalmente, será almacenada en un almacenamiento persistente en el sistema del usuario de una forma totalmente insegura. La amenaza aquí es que el artefacto podría ser utilizado posteriormente. La propiedad obligatoria de "sólo un uso" de los artefactos SAML garantiza que no podrán ser reutilizados por el navegador. Debido a los tiempos de vida cortos de los artefactos y de las afirmaciones obligatorias SSO es difícil robar un artefacto y reutilizarlo más tarde desde otro navegador.

Estudiado en profundidad el perfil SAML navegador/artefacto SSO pasemos ahora a analizar el otro posible tipo de perfil: *Browser/POST Profile of SAML*

Este perfil permite el mismo efecto de autenticación en un sitio Web destino de manera transparente que el esquema anterior pero esta vez sin el uso de un artefacto SAML en una URL de redirección sino mediante peticiones HTTP POST. El siguiente diagrama muestra las interacciones entre las partes en el perfil Browser/POST. Este perfil consiste de una

serie de dos interacciones, la primera entre un usuario equipado con un navegador Web y un sitio Web origen, y la segunda directamente entre el usuario y el sitio de destino.

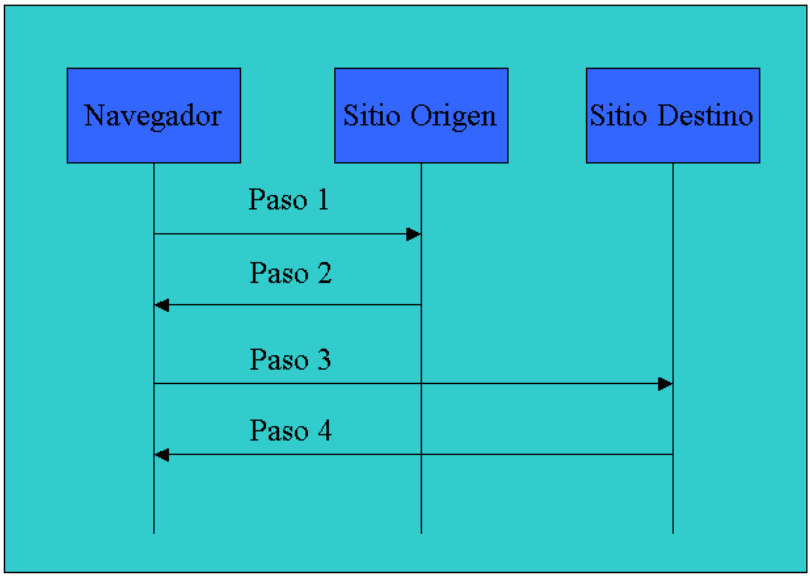

Ilustración 18. Pasos definidos por el perfil Browser/POST.

En el paso 1, el navegador del usuario accede al servicio de transferencia entre sitios Web ubicado en el host *https://<nombre-host-servicio-transferencia>*, pasándole la información sobre el recurso deseado (localizado en el sitio Web de destino) asociado en la URL. No existe una forma normativa para este paso 1 aunque se recomienda que la petición HTTP tenga la siguiente forma:

GET <path>?…TARGET=<Recurso>…<HTTP-Version>
<otros componentes HTTP 1.0 or 1.1>

El campo path indica los componentes del path de la URL del servicio de transferencia entre sitios Web ubicado en el sitio Web origen. Por su parte, el elemento …*TARGET=<Recurso>* permite transmitir la información sobre el recurso deseado.

En el paso 2, el sitio Web origen genera un formulario de datos HTML que contiene un mensaje de respuesta SAML el cual, a su vez, contiene una afirmación SSO.

Es de destacar que en este perfil, la URL utilizada para acceder al servicio consumidor de afirmaciones en el sitio Web destino es referenciada como la *URL del consumidor de las aseveraciones*.

La respuesta HTTP, ésta si es normativa, debe tener la siguiente forma:

<HTTP-Version> 200 <Frase con el motivo >
<otros componentes HTTP 1.0 or 1.1>

Donde la etiqueta <otros componentes HTTP 1.0 or 1.1> debe incluir el siguiente formulario HTML:

<Body>

<FORM Method="Post" Action="https://<URL al host consumidor de servicios ubicado en el sitio Web destino >" …>
<INPUT TYPE="hidden" NAME="SAMLResponse"
Value="B64(<respuesta>)">
…
<INPUT TYPE="hidden" NAME="TARGET" Value="<Recurso>">
</Body>
<host consumidor de la aseveración y path>

Este último componente debe proporcionar el nombre del host, el puerto y los componentes del path de una URL de un consumidor de aseveraciones localizado en el sitio Web destino. Sólo se debe incluir una respuesta SAML dentro del cuerpo del formulario HTML con el nombre de control *SAMLResponse.* Se pueden incluir múltiples afirmaciones en la respuesta y al menos una de ellas debe ser de tipo SSO. Además, sólo se puede incluir la descripción de un recurso con el nombre *TARGET*.

La notación B64 (<respuesta>) representa el resultado de aplicar la Codificación de Transferencia de Contenido a la respuesta tal y como se describe en el RFC 2045 y debería consistir de líneas codificadas con a lo sumo 76 caracteres cada una.

La respuesta y las aseveraciones SAML deben ser firmadas digitalmente. La confidencialidad e integridad de los mensajes debe ser mantenida en el paso 2. Además, la especificación recomienda que la URL del servicio de transferencia entre sitios Web sea protegida mediante SSL 3.0 o TLS

1.0 para evitar que las afirmaciones devueltas sean transportadas en texto plano y sean objetivo fácil para ataques de suplantación de personalidad utilizando afirmaciones válidas.

En el paso 3, el navegador envía el formulario que contiene la respuesta SAML mediante una petición HTTP al servicio consumidor de aseveraciones ubicado en el host *https://<nombre-host-servicio-consumidor-aseveraciones>*.

La petición HTTP debe incluir los siguientes componentes:

POST <path> <http-Version>
<Otros componentes HTTP 1.0 o 1.1 de la petición>

Donde <path> son los componentes del path de la URL del servicio consumidor de aseveraciones en el sitio destino y *<Otros componentes HTTP 1.0 o 1.1 de la petición>* consiste del conjunto de datos del formulario derivado del procesamiento realizado por el navegador sobre los datos del formulario recibido en el paso 2. En estos datos del formulario se debe incluir exactamente una respuesta SAML cuyo nombre de control sea *SAMLResponse* y una o muchas aseveraciones. Además, también debe incluir la descripción de un único recurso al que se hace referencia mediante el nombre *TARGET*. La respuesta SAML debe incluir el atributo *Recipient* con un valor establecido a *https://<nombre y camino del host consumidor de la afirmación>*. Recordar que este atributo es opcional e indica el único posible receptor

de la respuesta. Este atributo es muy útil contra ataques de re-envío de los mensajes hacia destinatarios no autorizados, una protección establecida como requisito indispensable en muchos perfiles de uso de SAML (como por ejemplo éste). Su valor es una URI que referencia al receptor. Al menos una de las afirmaciones SAML que están incluidas en la respuesta debe ser de tipo SSO.

El sitio Web destino debe garantizar una política que asegure que las afirmaciones SSO comunicadas por medio de este perfil sólo sean utilizadas una vez. O, lo que es lo mismo, no sean reutilizables. La consecuencia directa de este requisito es que el sitio de destino necesitará mantener y salvar el estado de las afirmaciones SSO que recibe de forma que pueda saber si está reutilizando o no alguna.

Por supuesto, se debe garantizar tanto la integridad como la confidencialidad del mensaje de petición HTTP del paso 3. La especificación recomienda el uso de SSL 3.0 o TLS 1.0 en la URL del servicio consumidor de aseveraciones. Si estas dos propiedades no estuvieran garantizadas, las afirmaciones transmitidas en el paso 3 se encontrarán accesibles en texto plano para cualquier atacante que podría entonces suplantar la identidad del sujeto de la aseveración. Cada declaración sobre un sujeto en las aseveraciones enviadas al sitio Web destino deben contar con un elemento *saml:SubjectConfirmation* en el cual el elemento hijo *ConfirmationMethod* tenga establecido el valor *urn:oasis:names:tc:SAML:1.0:cm:bearer*. Recordemos que las aseveraciones que incluyen un sujeto, deben describir a éste, bien

mediante un elemento *NameIdentifier* y un elemento un *SubjectConfirmation* o sólo con éste último. El elemento *SubjectConfirmation* debe contener información que permita al receptor de la aseveración verificar la identidad de su emisor.

En el último paso, el paso 4, el navegador del usuario recibe una respuesta HTTP que reflejará si el acceso al recurso solicitado ha sido autorizado o denegado. La especificación SAML no propone una forma normativa para la respuesta HTTP de este paso aunque señala que el sitio Web de destino debería proporcionar alguna forma de clarificación, como mensajes de error, en el caso en de que el acceso al recurso haya sido rechazado.

A continuación estudiaremos, al igual que lo hicimos con el perfil navegador/artefacto SSO de SAML, el conjunto de amenazas identificadas para este perfil así como las formas de mitigarlas o, en los mejores casos, solucionarlas.

La primera amenaza que está identificada es el caso en que un atacante intenta robar una aseveración. En este caso, si el atacante es capaz de copiar la respuesta SAML real, incluidas sus aseveraciones, entonces podrá construir un formulario POST válido suplantando de esta forma al usuario frente al sitio Web destino. La forma de solucionar esto, y como ya se ha dicho en las descripciones de los pasos 2 y 3, es garantizando la confidencialidad siempre que se envíe una respuesta de uno de los dos sitios Web y el navegador del usuario (mediante SSL 3.0 o TLS 1.0). En

el caso en que el atacante rompiera esta barrera de la confidencialidad se podrían incluir las mismas medidas adicionales que se describieron para este mismo tipo de ataque en la especificación del perfil navegador/artefacto SSO de SAML:

- Sincronización de los relojes de los sitios Web origen y destino.
- Valores *NotBefore* y *NotOnOrAfter* del elemento *Assertion* que acoten el tiempo de vida de las afirmaciones (normalmente establecido en unos pocos minutos).
- La comprobación por parte del sitio Web destino del periodo de validez de las afirmaciones que le llegan.

En el caso en que se reciba una declaración de autenticación con un elemento *saml:SubjectLocality* , el sitio Web destino podría verificar si la IP navegador frente a la IP contenido en dicho elemento.

Otro tipo posible de ataque es el denominado ataque MITM. En este caso podemos pensar que el sitio Web destino, que recibe las afirmaciones SAML, podría actuar de manera maliciosa y utilizarlas él mismo para suplantar la identidad del usuario original frente a otro sitio Web de destino. El nuevo sitio de destino creerá que el sitio malicioso es el sujeto de las afirmaciones.

La manera de prevenir estas situaciones es que el nuevo sitio Web de destino verifique el atributo *Recipient* de la respuesta SAML para asegurarse de que su valor coincide con *https://<nombre de host consumidor de aseveraciones>*. Como la respuesta debe estar firmada

digitalmente, el valor del recipiente no podrá ser alterado por un sitio Web destino malicioso.

El siguiente tipo de ataque que se puede estudiar es el forjado de las aseveraciones. Un usuario atacante podría crear aseveraciones o modificar alguna ya creada. El perfil navegador/POST requiere que la respuesta SAML que contiene las aseveraciones SAML se encuentre firmada y, de esta forma, se garantice tanto la integridad como la autenticación del mensaje. El sitio Web de destino debe verificar la firma y autenticar al emisor.

Otro ataque más, que también se contemplaba en el perfil artefacto/navegador, es aquel que guarda relación con la exposición del estado del navegador a posibles atacantes. Como el navegador se descarga las aseveraciones para luego enviarlas al servicio consumidor de aseveraciones, pasarán a estar disponibles como parte del estado del navegador ya que usualmente éste las almacenará en algún tipo de sistema persistente de manera no segura. La amenaza aquí es que la aseveración podría ser reutilizada en algún momento posterior. La contramedida para este caso es que las aseveraciones SSO deben tener establecido un tiempo de vida muy corto y los sitios de destino deben comprobar que las aseveraciones SSO no son re-enviadas desde el navegador.

6.2.14 Consideraciones de seguridad

En la siguiente sección se estudiarán una serie de consideraciones de seguridad que el documento *Security and Privacy Considerations for the OASIS Security Assertion Markup Language V1.1* (Lockhart et al., 2003) que forma parte del grupo documental de la especificación analiza con gran detalle.

Privacidad

Como ya hemos estudiado en secciones anteriores, SAML incluye la capacidad de construir afirmaciones sobre los atributos (incluida la identidad) o sobre autorizaciones concedidas a entidades autenticadas. En muchas situaciones se podría dar el caso de que las entidades o sujetos sobre los que se realizan las afirmaciones desearan mantener parte de la información oculta, ya sea de atributos o de autorización. Es más, en función al consumidor de la aseveración el sujeto podría querer incluir unas u otras declaraciones. Todas las partes que juegan su papel en la creación de declaraciones y emisión, transmisión o consumo de afirmaciones SAML deben de ser conscientes de que pueden existir estas restricciones potenciales de privacidad y deberían intentar resolverlas en la implementación de sus sistemas SAML.

Confidencialidad

Con toda seguridad, el aspecto más importante para asegurar la privacidad de las partes implicadas en una transacción SAML es garantizar la confidencialidad. En otras palabras ¿es posible transmitir la

información en una afirmación SAML desde el emisor hasta un cierto grupo selectivo de receptores, y sólo a este grupo, sin hacerla accesible a ninguna otra parte? Técnicamente es absolutamente factible transmitir la información de manera confidencial mediante mecanismos criptográficos. Pero es de destacar, que el simple hecho de hacer invisibles los contenidos de las afirmaciones puede no cubrir todos los requisitos de privacidad deseados. Existen muchos casos donde el simple hecho de que esté disponible la información de que cierto usuario (o una dirección IP) estuvo accediendo a un servicio podría constituir una brecha en la privacidad. Para evitar este tipo de "asaltos" a la privacidad se pueden emplear técnicas de interacción anónima (ver WS-Federation) que más tarde analizaremos.

Anonimato

En este apartado vamos a discutir el concepto de anonimato.

Actualmente no existen definiciones sobre el anonimato que satisfagan todos los posibles escenarios en los que este concepto puede ser utilizado. Una definición muy común sobre el término en cuestión guarda relación con un emisor de un mensaje. Según esta definición, el emisor será anónimo si el receptor no es capaz de conocer su identidad a partir del mensaje recibido.

Sin lugar a dudas esta definición es correcta para este caso en particular, aunque no sería completa debido a la posibilidad del receptor de acumular información a lo largo del tiempo y, tras sucesivas

interacciones con el emisor, tener una sólida descripción, no quizá sobre su identidad, pero sí sobre su conducta.
Existen dos términos que nos podrían ayudar en nuestra búsqueda de la definición de anonimato. El primer punto es pensar sobre el anonimato como ser o estar "dentro de un grupo" tal y como se expresa en el siguiente comentario extraído de "Anonymity, Unobservability, and Pseudonymity" (Pfitzmann & Köhntopp)**:**

"...Anonymity is the stronger, the larger the respective anonymity set is and the more evenly distributed the sending or receiving, respectively, of the subjects within that set is."

Esta noción es muy relevante en SAML debido al uso de las autoridades. Recordemos que SAML define tres posibles tipos de autoridades: Autoridad de Autenticación, Autoridad de Atributo y Autoridad Punto de Decisión de Política. Aunque cierto sujeto fuera "anónimo", aún sería identificable como un miembro del conjunto de posibles sujetos dentro del dominio de cierta autoridad SAML.

En el caso de que el usuario disponga de una agregación de atributos, el conjunto puede llegar a ser más pequeño. Por ejemplo, si el usuario es anónimo pero tiene asociado un atributo *studentInCourse6th@mit.edu.* Ciertamente, el número de estudiantes que están en sexto curso es mucho menor que el grupo de estudiantes de todo el MIT.

El término “nombre real” (true name) fue introducido por Vinge y popularizado por May para referirse a la identidad legal de un individuo.
Tal y como se expresan Dingledine, Freedman, y Molnar en el documento Freehaven (Dingledine, Freedman, & Molnar, 2000): El anonimato y el pseudónimo protegen la privacidad de la *ubicación* del usuario a la par que su *“nombre real”*.
La ubicación se refiere a la conexión física real al sistema. El término “nombre real” fue introducido por Vinge y popularizado por May para referirse a la identidad legal de un individuo. Saber el nombre real o la ubicación de cierto individuo nos podría permitir, asumiendo el rol de atacante, producirle todo tipo de daños.

La misma fuente, (Dingledine et al., 2000), nos lleva a la unificación de la noción de anonimato “dentro un conjunto” y a la capacidad de producir daños:

Podríamos decir que un sistema es parcialmente anónimo si un adversario puede acotar la búsqueda de la identidad de un usuario a un ‘conjunto de sospechosos’. Si el conjunto es lo suficientemente grande, sería prácticamente imposible para un atacante actuar como si alguno de los ‘sospechosos’ fuera culpable. Por otro lado, cuando el ‘conjunto de sospechosos’ es pequeño, la mera sospecha que tenga el adversario de que cierto individuo se encuentra dentro de ese conjunto, podría permitirle tomar acciones contra todos. Los sistemas SAML están limitados a ser, en el mejor de los casos, “parcialmente anónimos” debido al uso de las autoridades. Una entidad sobre la que se ha realizado una

afirmación pasa a formar parte del conjunto de entidades identificables por pura relación con la autoridad SAML. Las limitaciones del anonimato podrían ser muchos peores que una simple asociación con una autoridad dependiendo del empleo que se realice de los identificadores, puesto que la reutilización de los identificadores "pseudónimos" permiten la agregación de información que puede permitir la identificación.

Además, los usuarios de sistemas SAML pueden provocar brechas en su anonimato mucho peores debido a sus acciones.

A parte de la identidad legal, cualquier identificador "extra" asociado a un individuo puede ser considerado un pseudónimo. Incluso nociones como "poseedor de una clave" pueden ser consideradas como equivalentes de un pseudónimo cuando se asocia una acción (o conjunto de acciones) a un sujeto. También pueden ser considerados pseudónimos descripciones del tipo "el usuario que acabó de solicitar el acceso al objeto XYZ a las 23:24". Así, en cuanto a la capacidad que mencionábamos de producir daños por parte de un adversario, no existe diferencia si el usuario se describe con un identificador o si se describe por su conducta (por ejemplo, el uso de cierta clave o la ejecución de una acción). El punto que marca la diferencia es la frecuencia de uso que se haga de un pseudónimo en particular. Es decir, cómo de a menudo la misma identidad utilice el mismo pseudónimo. En (Pfitzmann & Köhntopp) **se** propone una taxonomía de pseudónimos comenzando desde los pseudónimos personales (como por ejemplo el 'Secretario de

Defensa'), hasta los pseudónimos que se utilizan solamente una vez. Estos últimos pueden proporcionarnos anonimato (que recordemos dentro de SAML se limita a ser un anonimato 'dentro de un grupo'). Cuanto más a menudo se utilice cierto pseudónimo, más estaremos reduciendo nuestro anonimato y más estaremos incrementando la probabilidad de que podamos ser atacados.

En otras palabras, la reutilización de un pseudónimo permite que se pueda asociar más información de identificación con el pseudónimo. Con el tiempo, esto provoca una acumulación de información que puede identificar inequívocamente la identidad asociada con el pseudónimo.

Hablemos un poco más sobre la relación entre la conducta de cierto usuario y el concepto de anonimato. Un usuario que cada martes a las 21:00 accede a una base de datos, que realiza una correlación de la longitud de los dedos con la duración de la vida, se convertirá rápidamente en poco anónimo. Dependiendo de los otros posibles tipos de conducta que ese usuario tenga, ésta puede llegar a ser "trazable" en función de si esa otra información de "identificación" puede ser acumulada o no.

Un usuario que rutinariamente compra un conjunto de productos de máquina de venta en red, ciertamente se está exponiendo fácilmente para ser atacado.

Veamos ahora cuáles son las implicaciones cuando manejamos entornos donde la privacidad resulta ser un requisito indispensable (o al menos esa es nuestra intención inicial). Las autoridades ubicadas en el origen (como por ejemplo autoridades de autenticación y autoridades de atributos) puede proporcionar un grado de "anonimato parcial" empleando identificadores que sólo se puedan utilizar una vez o mediante claves (para el caso del "poseedor de claves"). Este anonimato es a lo sumo parcial porque el Sujeto SAML se encuentra necesariamente confinado a un conjunto de Sujetos por su relación con la autoridad SAML. Este conjunto se podría reducir (reduciéndose así el anonimato) cuando se utiliza la agregación de atributos para acotar aún más el subconjunto de los posibles usuarios del sitio origen. Los usuarios que verdaderamente se preocupan sobre el anonimato deben tener la precaución de "disfrazar" o evitar patrones no usuales de conducta que pudieran servir para que los usuarios pierdan su anonimato con el paso del tiempo.

Otras consideraciones sobre la seguridad

La comunicación entre sistemas basado en computadoras se encuentra sujeta a una amplia variedad de amenazas que conllevan implícitamente una serie de riesgos. La naturaleza de estos riesgos varía desde el propio hecho de la comunicación hasta los entornos en los que se encuentran los sistemas comunicantes pasando por el propio medio de comunicación utilizado.

SAML tiene como propósito ayudar a los desplegadores a establecer contextos de seguridad a nivel de aplicación para las comunicaciones basadas en computadoras.

Asumiendo este rol, SAML resuelve el aspecto de "autenticación del interlocutor" dentro de la seguridad de las comunicaciones y, también, el aspecto de "uso no autorizado" perteneciente a los sistemas de seguridad. Algunas de las áreas que impactan ampliamente en la seguridad global de un sistema que utiliza SAML están explícitamente fuera del alcance del propio SAML. Algunas de ellas son:

Autenticación inicial

SAML permite crear declaraciones o afirmaciones sobre que cierto proceso de autenticación se ha llevado a cabo, pero no incluye como requisito ni especifica los procesos de autenticación en sí, tan sólo que se llevaron a cabo. Los consumidores de las afirmaciones de autenticación deberían ser cautos y no confiar ciegamente en estas afirmaciones a menos que conozcan los fundamentos sobre los cuáles fueron hechas. La confianza en las afirmaciones nunca debe exceder a la confianza que llevaron al interlocutor que realiza la afirmación a concluir sus afirmaciones.

Modelo de confianza

En muchos casos, la seguridad de una conversación SAML dependerá del modelo de confianza subyacente, el cual está típicamente basado en una infraestructura de gestión de claves (por ejemplo, PKI o clave secreta).

Por ejemplo, los mensajes SOAP que incorporan firmas digitales utilizando los mecanismos descritos por la especificación W3C XML Digital Signature serán seguros en tanto en cuanto las claves utilizadas para generar la firma sean de confianza. Si éstas estuvieran comprometidas la firma digital hecha sobre cierto conjunto de afirmaciones SAML no serían, desde el punto de vista de seguridad, de confianza. No detectar que las claves han sido comprometidas o que ciertos certificados han sido revocados podría provocar una brecha en la seguridad. Incluso, un fallo por no requerir un certificado podría abrir la puerta para ataques de suplantación. Es bien conocido que el establecimiento de una infraestructura PKI no resulta trivial y debe ser implementada correctamente con el fin de que las capas construidas sobre ella sean seguras y de confianza.

Por tanto, y a modo de resumen, debemos especificar que SAML no entiende de procesos de autenticación, solamente de si se han llevado a cabo o no con éxito, y que la confianza que podemos tener en un sistema que utiliza SAML está directamente relacionada con la confianza que podamos tener en las infraestructuras subyacentes a los sistemas que lo soportan.

A continuación estudiemos el modelo de amenaza para los sistemas que utilizan SAML. El modelo de amenaza general de Internet descrito por el IETF supone las líneas básicas para el modelo de amenaza de SAML. Asumiremos aquí que los dos o más puntos de comunicación o interlocutores de una transacción SAML no se encuentran

comprometidos pero que el atacante posee un control completo sobre el canal de comunicaciones.

Además, debido a la naturaleza de SAML como sistema de autenticación de múltiples partes y como protocolo intercambio de declaraciones de autorización, se deben considerar aquellos casos en los que uno o más de los interlocutores de una transacción SAML legítima, que operan legítimamente dentro de su rol asignado para la transacción, intentan utilizar de manera maliciosa en posteriores transacciones la información conseguida en transacciones previas.

En todos los casos, los mecanismos locales que los sistemas utilizarán para decidir si generar o no ciertas aseveraciones se encuentran fuera del propósito de esta discusión. La consecuencia directa de cerrar el ámbito del modelo de amenazas de SAML de esta manera es que la seguridad de un sistema basado en recibir afirmaciones como entrada es como mucho tan buena como la seguridad del sistema utilizado para generar esas afirmaciones. Cuando se debe determinar en qué emisor se debe confiar, especialmente en aquellos casos en los que las afirmaciones SAML serán utilizadas como entrada para la toma de decisiones de autenticación o autorización, el riesgo de comprometer la seguridad surgido por procesar afirmaciones falsas, aunque correctamente emitidas, es muy alto.

Veamos algunas técnicas de seguridad que podrán servirnos de ayuda y por tanto habría que tener en cuenta cuando desplegamos sistemas basados en SAML.

Autenticación

La autenticación significa aquí la capacidad que debe tener una parte, implicada en una transacción, de autenticar a todas y cada una de las otras partes envueltas en la misma transacción. La autenticación podría ser de un solo sentido o bidireccional. Se podrían hacer dos posibles distinciones en cuanto a los tipos de autenticación que se pueden dar:

- **Sesión activa.** La autenticación no persistente es ofrecida por el canal de comunicaciones utilizado para transportar los mensajes SAML. Esta autenticación podría ser unilateral, desde el iniciador de la sesión hacia el receptor, o bilateral. Este método específico estará determinado por el protocolo de comunicaciones utilizado. Por ejemplo, el uso de un protocolo de red seguro, como el propuesto por el RFC 2246 o el Protocolo de Seguridad de IP IPSec, habilita al emisor del mensaje a autenticar al destinatario dentro del entorno TCP/IP.
- **Autenticación a nivel de mensaje.** La especificación W3C XML Signature y la especificación OASIS Web Services Security proporcionan un método de creación de autenticación persistente que está altamente acoplado al documento. Este método no garantiza de manera independiente que el emisor del mensaje es de hecho el firmante (de hecho, en muchos casos donde existan intermediarios, éste no será el caso).

Confidencialidad

La confidencialidad significa que los contenidos de un mensaje pueden ser solamente leídos por los destinatarios que estén autorizados y por

nadie más que tenga acceso físico al mensaje. Podemos distinguir dos tipos de maneras de asegurar la confidencialidad de los mensajes:

- **Confidencialidad en tránsito.** Utilizar un protocolo de red seguro como los mencionados RFC 2246 o IPSec proporcionan una confidencialidad volátil de un mensaje cuando es transmitido entre dos nodos.
- **Confidencialidad a nivel de mensaje.** La especificación W3C XML Encryption ofrece una forma de cifrar de manera selectiva documentos XML. Este método de cifrado proporciona confidencialidad persistente y selectiva de la información contenido dentro de un mensaje XML.

Integridad de los datos

La integridad de los datos supone la capacidad de confirmar que cierto mensaje recibido en un extremo tiene la misma forma y no ha sufrido alteración alguna respecto a cuando fue enviado por el nodo emisor. La integridad de los datos puede estudiarse a dos niveles:

- **Integridad de los datos en tránsito.** Al igual que ocurría con la autenticación y la confidencialidad, el uso de un protocolo de red seguro como RFC 2246 o IPSec podría ser suficiente para proporcionar integridad de los CRCs de los paquetes transmitidos vía la conexión de red.
- **Integridad a nivel de mensaje.** Al igual que ocurría con la autenticación a nivel de mensaje, se puede hacer uso de la especificación W3C XML Digital Signature para saber con certeza si cierto mensaje es o no íntegro.

Gestión de las Claves

La seguridad de los sistemas que tienen la capacidad de proporcionar autenticación, integridad, y confidencialidad mediante firmas digitales y técnicas criptográficas está relacionada de manera directa con la seguridad de los sistemas de gestión de claves que utilizan.

Se asume que, si los sistemas basados en claves van a ser utilizados para conseguir autenticación, integridad de los datos y no repudio, se deberán establecer ciertas medidas de seguridad que garanticen que el acceso a las claves no está disponible para actores no autorizados. En general, el acceso a las claves debería ser autorizado al conjunto mínimo de entidades posibles (particularmente importante para claves corporativas u organizacionales) y deberían estar protegidas con una palabra de paso u otros medios. Por supuestos, se deben aplicar las precauciones de seguridad estándar (no apuntar en un papel la palabra de paso, no abandonar el ordenador con acceso a la clave, y medidas así).

Por otro lado, para que un sistema basado en claves sea utilizado para llevar a cabo una autenticación, debe existir algún tipo de vinculación de confianza entre la entidad posesora y la clave. La verificación de una firma digital de un documento puede determinar si el documento no ha sido alterado desde que fue firmado, y que fue firmado por una clave dada. Sin embargo, no hay forma de confirmar que la clave utilizada es realmente la clave de un individuo específico. Esta vinculación “clave con individuo” debe ser siempre establecida. Las soluciones comunes incluyen directorios locales que, o bien almacenan directamente los

identificadores junto con sus claves, lo que es fácil de entender pero difícil de mantener, o bien almacenan certificados. Los certificados, que son en esencia vinculaciones firmadas de una identidad con una clave resultan ser una solución particularmente poderosa al problema, aunque no se encuentra exenta de los habituales inconvenientes. Para cada consumidor de una firma, se debe identificar un conjunto de Autoridades de Certificación raíces de confianza, que contesten a la cuestión "¿En quién confío para realizar declaraciones que vinculan a un identificador con una clave?". La verificación de la firma se convierte, en primer término, en un proceso de verificación de la firma (para determinar que la firma fue realmente realizada por la clave en cuestión y que el mensaje no ha sufrido cambio alguno) y, en un segundo término, en un proceso de verificación de la cadena de certificación (para determinar que la clave está vinculada con la identidad correcta). Además, cuando se hace uso de los certificados se deben tener en cuenta ciertas consideraciones para garantizar que la vinculación es válida, como por ejemplo que el certificado no haya caducado. Un certificado tiene normalmente un tiempo de vida máximo y que cuando se supera deja de ser oficialmente válido teniendo que iniciar el proceso oportuno para su renovación. Si durante el tiempo de vida de un certificado la clave se viera comprometida en algún momento, entonces la relación "identidad, clave" contenida en el certificado se convierte en no válida pese a que el certificado aún no haya caducado. También, un certificado puede depender a menudo de asociaciones que podrían finalizar antes de que su tiempo de vida expire (por ejemplo, un certificado que dejara de ser válido cuando cierto empleado se marcha). Este problema se resuelve

mediante las Listas de Revocación de Certificados (CRLs), que son listas de certificados emitidos por cierto CA y que han sido revocados desde su emisión. Otra solución es el Online Certificate Status Protocol (OCSP) que define un método para invocar a los servidores con el propósito de preguntar si cierto certificado es o no válido. Alguna de esta misma funcionalidad está incorporada en los niveles más altos de la especificación W3C XML Key Management System, la cual permite realizar peticiones de claves "válidas". Un sistema de gestión de claves robusto es una herramienta muy sólida y útil pera muy compleja de construir. La verificación de una firma acaba siendo un proceso de tres pasos: verificación de la vinculación de la clave con el documento; verificación de la vinculación de la identidad con la clave; y finalmente verificación de la validez actual de la relación entre la clave y el documento.

6.2.15 Dependencias con otras especificaciones

Como hemos visto a lo largo del estudio de la especificación SAML:

- Utiliza firmas digitales como elementos hijo del elemento *Assertion* con el fin de garantizar la integridad y la identidad del emisor de la aseveración.
- Utiliza el tipo de dato **dateTime** contenido en la especificación W3C XML Schema como tipo de dato de los atributos del *NotBefore* y *NotOnOrAfter* del elemento *Conditions*.
- En el tipo de declaración *SubjectStatement,* se define un elemento hijo *Subject*, que, a su vez puede:

(1) Contener un elemento hijo *NameIdentifier* y opcionalmente un elemento hijo *SubjectConfirmation*.
(2) Sólo contener un elemento *SubjectConfirmation*.

- El elemento *SubjectConfirmation*, cuyo propósito recordemos es proporcionar información suficiente para que el servicio SAML que reciba la aseveración pueda autenticar al sujeto de la misma, contiene:
  - Uno o más elemento *ConfirmationMethod*, indicando el método de verificación que puede ser utilizada para verificar la información contenida en el elemento *SubjectConfirmation*.
  - Opcionalmente puede contener un elemento *SubjectConfirmationData* que debe contener información adicional necesaria para que el servicio SAML que verifique la aseveración puede consultarlo.
  - Además, y es aquí donde viene la relación con otra especificación, puede contener un elemento *ds:KeyInfo* importado del espacio de nombres definido por la especificación *XML Digital Signature*. Este elemento contendrá una clave cuya propiedad es del sujeto de la aseveración.

- Utiliza el tipo de dato *QName* definido en el esquema W3C XML Schema para el tipo de dato del atributo *AuthorityKind* del

elemento *AuthorityBinding* a su vez elemento hijo del elemento *AuthenticationStatement*.

- El tipo de dato abstracto *RequestAbstractType* padre del elemento *RequestType* (ver estudio del protocolo petición/respuesta SAML) permite opcionalmente reflejar un elemento *ds:Signature* con el fin de posibilitar al solicitante reflejar firmas digitales de las peticiones que realice.
- El tipo de dato abstracto *ResponseAbstractType*, base para los tipos de respuestas, también especifica que opcionalmente se puede incluir una firma del contenido de la respuesta SAML mediante el elemento *ds:Signature*.

6.3 XACML

Organización	OASIS
Estado	Liberada
Versión	1.1 (7 Agosto 2003)

6.3.1 Introducción

En este apartado vamos a realizar una introducción a XACML (eXtensible Authorization Markup Language).

Siendo breves, podemos definir XACML como un lenguaje, basado en XML, para definir políticas de control de acceso. Esto significa que proporciona una sintaxis para gestionar el acceso a los recursos por parte de ciertos sujetos que desean realizar ciertas operaciones sobre ellos.

XACML es un estándar cuya responsabilidad recae en el consorcio OASIS, que describe tanto un lenguaje de políticas como un protocolo petición/respuesta para el control de las decisiones de autorización. Ambos, lenguaje y protocolo, están definidos en XML. El lenguaje de políticas XACML se utiliza para describir requisitos de control de acceso generales, y tiene unos puntos de extensión estándar que permiten la definición de nuevas funciones, tipos de datos, combinaciones lógicas, etc. El protocolo petición/respuesta nos permite componer una petición

para solicitar si cierta acción debería, o no, ser permitida, además de ofrecernos formas de interpretar las decisiones de control. La respuesta siempre incluye una contestación que refleja si la petición debería ser o no permitida utilizando uno de los siguientes cuatro valores: *Permit*, *Deny*, *Indeterminate* (no se pudo tomar una decisión porque ocurrió un error o fue omitido algún valor requerido) o *NotApplicable* (la petición no puede ser contestada por este servicio). Estos valores se encuentran predefinidos por la propia especificación.

XACML define una arquitectura de servicios que debe implementar toda infraestructura de políticas completas:

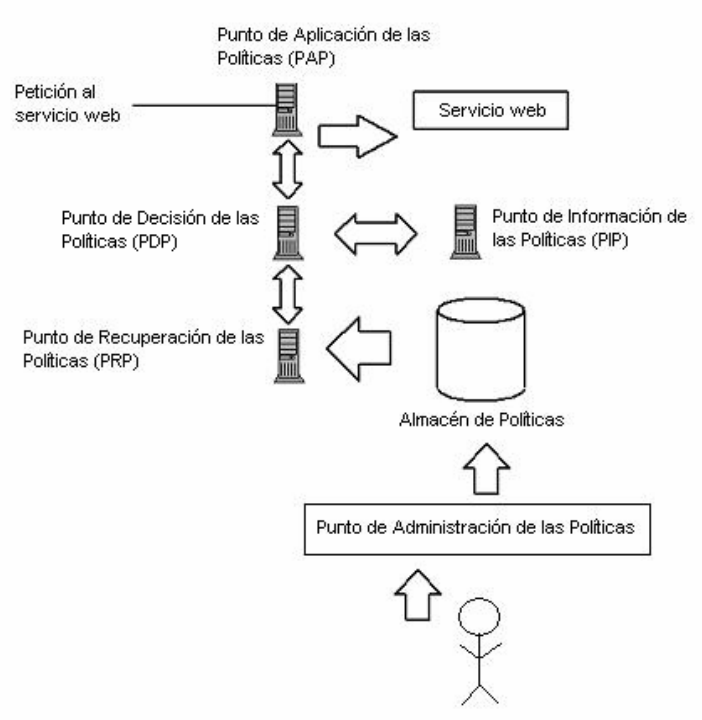

Ilustración 19. Arquitectura de servicios de políticas XACM L.

En la figura se muestran los servicios, y los diálogos entablados entre los mismos, que intervienen en la toma de decisión de autorizar el acceso de una petición XACML sobre SOAP a cierto servicio Web.

La petición SOAP que viaja hacia el servicio Web es interceptada por el PEP (Policy Enforcement Point) cuya tarea es forzar la ejecución de la lógica de políticas sobre la petición. El PEP pregunta al PDP (Policy Decision Point) mediante un mensaje SAML de autorización. El PDP debe decidir si se autoriza o no la petición. Para ello, y en el caso de que

no disponga de las políticas necesarias, realiza una petición XACML de políticas al PRP (Policy Retrieval Point) que le devuelve las políticas en formato XACML necesarias para evaluar la petición. Tras recibir las políticas el PDP podría necesitar obtener información, en forma de atributos, sobre el sujeto o su entorno, con el fin de poder evaluar la política. El PDP solicitará esta información al PIP (Policy Information Point) que le devolverá las afirmaciones de atributos SAML necesarias. Finalmente el PDP evaluará la política y si la petición es autorizada devolverá una afirmación de autorización SAML al PEP. Finalmente la petición alcanzará el servicio Web destino.

La situación típica es que alguien quiera realizar cierto conjunto de acciones sobre un recurso. Realizará una petición al servicio que realmente protege el recurso (como un sistema de ficheros o un servidor Web), que se denomina Punto de Aplicación de la Política o PEP (Policy Enforcement Point). El PEP formará una petición basada en los atributos del solicitante, el recurso en cuestión, la acción y otra información que pertenezca a la petición. El PEP enviará la petición al PDP que deberá ser quién evalúe si la petición debe ser aceptada o denegada.

Existen muchos lenguajes específicos de aplicación o propietarios que ya realizan las mismas acciones, aunque XACML posee varios puntos a su favor:

- Es un estándar. Utilizar un lenguaje estándar supone utilizar algo que ya ha sido revisado por una comunidad de expertos y usuarios muy grande, evitándonos tener que pensar sobre los todos los

puntos, incluidos los más críticos, implicados en diseñar nuestro propio lenguaje. Además, como XACML se está utilizando cada vez más, será más sencillo interactuar con otras aplicaciones utilizando el mismo lenguaje.

- Es genérico. Esto significa que en vez de intentar proporcionar control de acceso para un entorno en particular o un tipo de recurso específico, puede ser utilizado dentro de cualquier entorno. Una política puede ser escrita de forma que puede ser utilizada por muchas aplicaciones diferentes, y como se hace uso de un lenguaje común, la gestión de la política se simplifica enormemente.
- Es distribuido. Esto significa que una política puede ser escrita de forma que puede referenciar otras políticas ubicadas en localizaciones arbitrarias. El resultado es que en vez de tener que gestionar una única política monolítica, diferentes agentes o grupos pueden gestionar sub-piezas de las políticas de manera apropiada, y XACML conoce cómo combinar correctamente los resultados de estas diferentes políticas para tomar una única decisión.
- Es poderoso. Pese a que existen múltiples maneras de extender el lenguaje base, muchos entornos no necesitarán hacerlo. El lenguaje estándar ya soporta una amplia variedad de tipos de datos, funciones, y reglas para combinar los resultados de las distintas políticas. Además, ya existen muchos grupos estándares que se encuentran trabajando sobre extensiones y perfiles que permitirán “enganchar” XACML con otros estándares como

SAML o LDAP, cosa que incrementará el número de formas en el que XACML puede ser utilizado.

Para dar una idea mejor de cómo todos estos aspectos encajan de manera conjunta, realizaremos una breve discusión de la estructuras de las políticas XACML, lo que demostrará muchas de las propiedades estándar del lenguaje. Debemos destacar que XACML es un lenguaje rico, de forma que lo que se mostrará en este apartado será solamente algunas de sus características.

En la siguiente sección, y como hemos venido haciendo en todas las especificaciones que estamos estudiando en este documento, realizaremos un análisis del estado de la especificación y de su contenido.

Constructores de alto nivel: Policy y PolicySet

La raíz de todas las políticas XACML es un elemento *Policy* o *PolicySet*. Un elemento *PolicySet* es un contenedor que puede contener otros elementos *Policy* o *PolicySet*, así como referencias a políticas que se encuentran en ubicaciones remotas. Un elemento *Policy* representa una única política de control de acceso, expresada a través de un conjunto de reglas *Rules*. Cada documento de política XACML contiene exactamente un elemento raíz *Policy* o un *PolicySet*.

Puesto que un elemento *Policy* o *PolicySet* puede contener múltiples *Rules*, cada una de ellas podrían evaluarse a diferentes decisiones de

control de acceso. XACML necesita alguna forma de reconciliar las decisiones que produce cada regla.

Esto se realiza mediante una colección de elementos que permiten reflejar una combinación de algoritmos *Combining Algorithms.* Cada algoritmo representa una forma distinta de combinar las múltiples decisiones para alcanzar una decisión única. Existen Algoritmos de Combinación de Políticas (utilizados por los elementos *PolicySet*) y Algoritmos de Combinación de Reglas (utilizado por una política *Policy*). Estos Algoritmos de Combinación se utilizan para construir incrementalmente políticas complejas y, mientras existen un total de siete algoritmos estándar, esto no impedirá que podamos construir nuestro propio "juego" de algoritmos adecuados a nuestras necesidades.

Objetivos y Reglas

Parte de lo que un servicio PDP (Policy Decision Point) de XACML necesita hacer es buscar una política que sea aplicable a una petición dada. Para conseguir esto, XACML proporciona otra característica denominada *Target*. Un *Target* es básicamente un conjunto de condiciones simplificadas para el sujeto (elemento *Subject*), recurso (elemento *Resource*) o la acción (elemento *Action*) que deben ser expresadas por un *PolicySet*, *Policy* o *Rule* de forma que puedan ser aplicadas sobre una petición dada.

Es decir, para poder determinar si cierta política o conjunto de políticas, o si cierta regla aplica a la petición XACML recibida se deberá analizar

el contenido del elemento *Target* que contiene el sujeto, el recurso y la acción sobre la que aplican.

Las condiciones utilizan funciones “boleanas” para comparar los valores encontrados en una petición con aquellos incluidos en un *Target*. Si todas las condiciones del *Target* se ven cumplidas, entonces su elemento *PolicySet*, *Policy* o *Rule* asociadas es aplicable a la petición. Además de ser una manera de comprobar la aplicabilidad, la información del *Target* también proporciona una forma de “indexar” las políticas, lo que resulta muy útil si poseemos muchas políticas y necesitamos realizar una búsqueda rápida a través de todas ellas para ver cuál es la que aplica.

Por ejemplo, una política representada mediante un solo elemento *Policy* podría contener un *Target* que solamente aplicará a una petición para un servicio específico. Cuando llegue una petición de acceso a ese servicio, el PDP sabrá donde ir a buscar las políticas que podrían aplicar a esa petición porque las políticas se encuentran indexadas basadas en las restricciones de sus *Target*.

Una vez se encuentra una política que aplica a la petición, se procede a evaluar las reglas *Rule* que contiene. Como ya hemos dicho, XACML se basa en políticas que contienen reglas. El resultado de combinar los resultados de evaluar cada una de las reglas contenidas en la política será el que refleje si, para esa política, la petición puede ser autorizada o no.

Una política puede tener cualquier número de reglas *Rule* las cuales contienen la lógica principal de una política XACML. El corazón de la mayor parte de las reglas es un elemento *Condition*, que es una función "boleana". Si la condición reflejada en el elemento *Condition* se evalúa como verdadero, entonces es devuelto el valor *Effect* de la regla (un valor 'Permit' o 'Deny' que está asociado con la evaluación con éxito de la regla). La evaluación de una condición también puede resultar en error (devolviendo entonces el valor 'Indeterminate') o en el descubrimiento de que esa *Condition* no aplica a la petición (devolviendo en este caso el valor 'NotApplicable'). Una condición *Condition* puede ser bastante compleja y puede estar construida a partir de un conjunto arbitrario de atributos y funciones "boleanas". En resumen, el corazón las políticas XACML son las reglas que las componen, y el corazón de las reglas las condiciones que contienen.

Atributos, Valores de Atributos y Funciones

La divisa con la que XACML trata son los atributos. Los atributos son valores con nombre de tipos conocidos que podrían ser cosas como un identificador de un emisor o una fecha y hora de emisión. Específicamente, los atributos son características del sujeto, del recurso, de la acción o del entorno para el cual la petición de acceso ha sido realizada. Un nombre de usuario, el fichero al que se quiere acceder y la hora del día son todos posibles valores de atributos. Cuando se envía una petición XACML desde un PEP a un PDP, ésta se compondrá básicamente de atributos que serán comparados con los valores de los

atributos en una política, pudiéndose llevar a cabo la toma de decisión de acceso.

Una política resuelve los valores de los atributos procedentes de una petición, o de alguna otra fuente (Punto de Recuperación de Información), mediante dos mecanismos: el elemento *AttributeDesignator* y el elemento *AttributeSelector*. El elemento *AttributeDesignator* permite a la política especificar un atributo dado un nombre, un tipo y, opcionalmente, un emisor del atributo. El PDP entonces buscará el valor de dicho atributo en la petición, o en cualquier otro lugar en el caso de que no se encontraran valores coincidentes en la petición. Existen cuatro tipos de designadores, uno para cada uno de los posibles tipos de atributos de una petición: *Subject*, *Resource*, *Action* y *Environment*. Puesto que los atributos *Subject* pueden descomponerse en subcategorías, se pueden especificar también elementos *SubjectAttributeDesignators* como una categoría en la que buscar los valores.

Por su parte, el elemento *AttributeSelectors* permite a una política buscar los valores de los atributos mediante una petición XPath.

Ambos elementos, *AttributeDesignator* y *AttributeSelector* pueden retornar múltiples valores (ya que podrían coincidir varios atributos en una petición) y, por lo tanto, XACML proporciona un tipo de atributo especial denominado *Bag* (Bolsa). Los elementos *Bags* son una colección desordenada que permiten duplicados, y son siempre los que los

designadores o los selectores devuelven, incluso si solamente coincidiera un valor. En el caso en que no se dieran coincidencias, se devolverá un elemento *Bag* vacío.

Una vez se devuelve un elemento *Bag*, que recordemos contiene el valor de una serie de atributos, se necesitará comparar estos valores de alguna manera con los valores esperados para poder tomar así las decisiones de acceso correctas. Esta comparación de los valores de los atributos devueltos con los valores esperados se realiza mediante un poderoso mecanismo de funciones proporcionado por la propia especificación XACML. Las funciones pueden trabajar sobre cualquier conjunto de valores de atributo y pueden devolver cualquier tipo de valor de atributo que esté soportado en el sistema. Las funciones pueden estar anidadas, de forma que podemos tener funciones que operen sobre la salida de otras funciones, pudiendo ser esta jerarquía arbitrariamente compleja. Además, se pueden escribir funciones propias que enriquezcan el lenguaje para expresar condiciones de acceso.

Un punto que debe tenerse en cuenta cuando se construyen estas jerarquías de funciones es que la mayoría de las funciones están definidas para trabajar sobre tipos específicos (como cadenas de caracteres o enteros) mientras que los designadores y los selectores siempre retornan valores *Bag*. Para manejar esto, XACML define una colección de funciones estándar de la forma *"[tipo]-one-and-only"* las cuales aceptan una bolsa de valores del tipo especificado y retorna un único valor si existiera un solo elemento en la bolsa, o un error si hay cero o múltiples

valores en la bolsa. Esta es una de las funciones más comunes que veremos en los elementos *Condition*. Las funciones *[tipo]-one-and-only* no son necesarias en los *Targets* ya que el PDP aplica automáticamente la función de "matching" para cada elemento del elemento *Bag*.

Ejemplo de Política

A continuación mostramos un ejemplo sencillo de un elemento *Policy* que utiliza algunas de las propiedades o características descritas en esta introducción a XACML.

Como se puede observar en el fragmento XACML presentado más abajo, el elemento *Target* de esta política expresa que la política sólo aplica a aquellas peticiones que estén dirigidas al servicio con nombre "ServidorEjemplo". La política tiene una regla (elemento *Rule*) con un *Target* que requiere una acción de "login" y una condición (elemento *Condition*) que aplica solamente si el sujeto (elemento *Subject*) está intentando hacer "login" entre las 9:00am y las 5:00pm. Se debe destacar que este ejemplo puede ser extendido para incluir otras reglas para diferentes acciones.

Si la primera regla aquí mostrada no aplicara, entonces se puede utilizar una regla por defecto que siempre retorne 'Deny', denegando el acceso a toda aquella petición para la que explícitamente no se haya concedido el acceso.

```
<Policy PolicyId="SamplePolicy"
      RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
      algorithm:permit-overrides">
   <!--
Esta política sólo aplica a peticiones dirigidas al ServidorEjemplo
     -->
   <Target>
      <Subjects>
       <!—Este política política aplica a cualquier sujeto que realice la petición -->
           <AnySubject />
        </Subjects>
      <Resources>
        <!—El recurso al que aplica esta política es aquel cuyo atributo resource-id
              sea igual (comparación con la función string-equal¨) a la cadena
              ‘ServidorEjemplo’ -->
        <ResourceMatch
              MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
                DataType="http://www.w3.org/2001/XMLSchema#string">S
                ervidorEjemplo</AttributeValue>
            <ResourceAttributeDesignator
                DataType="http://www.w3.org/2001/XMLSchema#string"
                AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-
                id" />
          </ResourceMatch>
        </Resources>
      <!—Cualquier acción es válida para que esta política aplique -->
      <Actions>
           <AnyAction />
        </Actions>
     </Target>
   <!-- A continuación del Target de la política se encuentra el conjunto de reglas que
      definen las semánticas de control de acceso .
   Esta primera regla determina si se debe permitir o no realiza login al sujeto de la
      petición -->

   <Rule RuleId="ReglaLogin" Effect="Permit">
      <!— Esta regla sólo aplicará si la acción solicitada es login -->
        -->
      <Target>
         <Subjects>
             <AnySubject />
          </Subjects>
         <Resources>
             <AnyResource />
          </Resources>
         <Actions>
            <ActionMatch
                MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
                equal">
               <AttributeValue
                   DataType="http://www.w3.org/2001/XMLSchema#string"
                   >login</AttributeValue>
```

```
                <ActionAttributeDesignator
                    DataType="http://www.w3.org/2001/XMLSchema#string"
                    AttributeId="ServerAction" />
            </ActionMatch>
        </Actions>
    </Target>

    <!—Tras definir el ámbito de aplicabilidad de la regla, se define las condiciones
       de la misma que la hacen evaluar al valor del atributo Effect del elemento
       Rule.
    Como veremos esta regla sólo permite realizar la acción de login si se encuentra
       entre las 9:00 a.m. y las 5:00 p.m -->

    <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-
              greater-than-or-equal">
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-
                  one-and-only">
                <EnvironmentAttributeSelector
                    DataType="http://www.w3.org/2001/XMLSchema#time"
                    AttributeId="urn:oasis:names:tc:xacml:1.0:environment:cur
                    rent-time" />
            </Apply>
            <AttributeValue
                DataType="http://www.w3.org/2001/XMLSchema#time">09:
                00:00</AttributeValue>
        </Apply>
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-less-
              than-or-equal">
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-
                  one-and-only">
                <EnvironmentAttributeSelector
                    DataType="http://www.w3.org/2001/XMLSchema#time"
                    AttributeId="urn:oasis:names:tc:xacml:1.0:environment:cur
                    rent-time" />
            </Apply>
            <AttributeValue
                DataType="http://www.w3.org/2001/XMLSchema#time">17:
                00:00</AttributeValue>
        </Apply>
    </Condition>
  </Rule>
 <!--En este espacio se podrían incluir más reglas -->

 <!--  Finalmente se define una regla que establece la política por defecto -->

   <Rule RuleId="FinalRule" Effect="Deny" />
</Policy>
```

Con este ejemplo terminamos este apartado cuyo propósito ha sido introducir el lenguaje XACML, identificando su propósito y sus elementos más relevantes.

6.3.2 Motivación

XACML es un estándar que proporciona una respuesta a la demanda del mercado de un sistema genérico de creación y ejecución de políticas de seguridad. Un sistema de políticas de seguridad de una gran corporación tendrá posiblemente un gran número de elementos y de puntos de aplicación de las políticas que gobiernan sus recursos. Los elementos de una política podrán ser gestionados por los departamentos de Sistemas de Información, Recursos Humanos, Servicios Jurídicos, etc. Por otro lado, las políticas podrían ser aplicadas en multitud de entornos como en los puntos de acceso a la intranet, en el servicio de correo, en los sistemas de acceso remoto, etc.

En la realidad, la práctica más común es gestionar cada una de las políticas en cada uno de los puntos de aplicación de políticas para así conseguir un sistema global de políticas de seguridad más preciso. En consecuencia, parece muy poco probable, y sería muy caro, intentar proponer la modificación de las políticas de seguridad de una gran corporación. Por lo tanto, resulta muy complicado obtener una visión consolidada de los puntos de control de seguridad que están siendo empleados y que se encuentran repartidos por todo el sistema. Al mismo tiempo existe una presión creciente procedente de los propios gobiernos y administraciones sobre los consumidores y reguladores para que

demuestren que están aplicando una serie de normas a la hora de proteger la información de la empresa y de los consumidores.

Por todos estos motivos, existe una necesidad apremiante de acordar un lenguaje común que permita expresar políticas de seguridad. XML resulta ser la elección más natural para la definición de este lenguaje común gracias a la facilidad de extensión que permite su sintaxis y semántica y a su generalizada aceptación por los mayores productores y fabricantes.

6.3.3 Contexto XACML

XACML tiene como propósito ser adecuado para una variedad de entornos de aplicación. El lenguaje fundamental se aísla del entorno de aplicación mediante el contexto XACML, tal y como se muestra en la figura 20 (en la cual el alcance de la especificación XACML se indica por el área localizada dentro del círculo).

El contexto XACML está definido en un esquema XML, describiendo una representación canónica de las entradas y de las salidas del PDP, que recordemos es el Punto de Decisión de las Políticas. Los atributos referenciados por una instancia de una política XACML podrían encontrarse en la forma de expresiones XPath (elemento *AttributeSelector*) aplicadas sobre el contexto, o designadores de atributos (*AttributeDesignator*) que identifiquen el atributo por el sujeto, recurso, acción o entorno y su identificador. La implementación debe convertir las representaciones de los atributos en el entorno de la

aplicación (p.e.: SAML, J2SE, CORBA, etc.) y las representaciones de los atributos en el contexto XACML. La manera de lograr esta correlación no está descrita en la especificación XACML. En algunos casos, como por ejemplo SAML, esta conversión se podría realizar de manera automática mediante el uso de transformaciones XSLT.

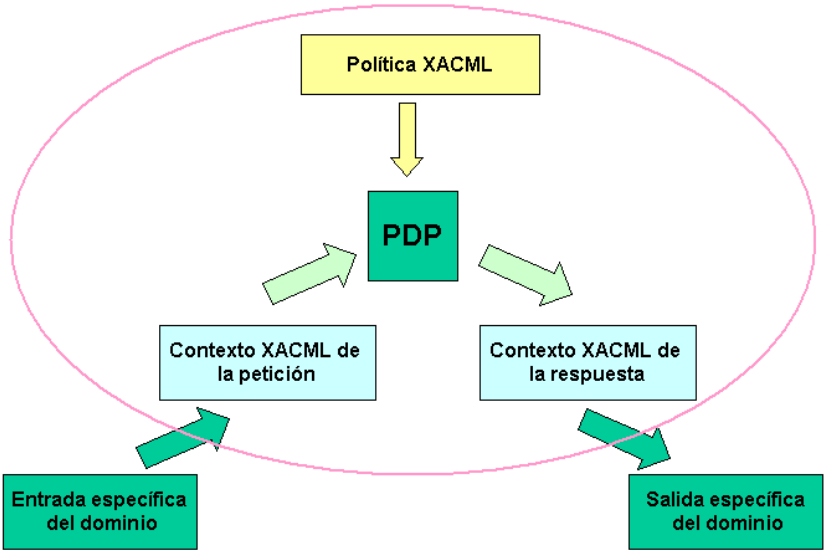

Ilustración 20. Contexto XACML.

6.3.4 Modelo del lenguaje de políticas

El estándar XACML describe un modelo que muestra los elementos básicos, así como sus relaciones, de las políticas que define. Tal y como vimos en la sección introductoria de esta especificación los elementos

básicos utilizados para definir políticas mediante XACML son ***PolicySet***, ***Policy*** y ***Rule***, ***Policy***.

Esos elementos serán descritos con más detalles a continuación:

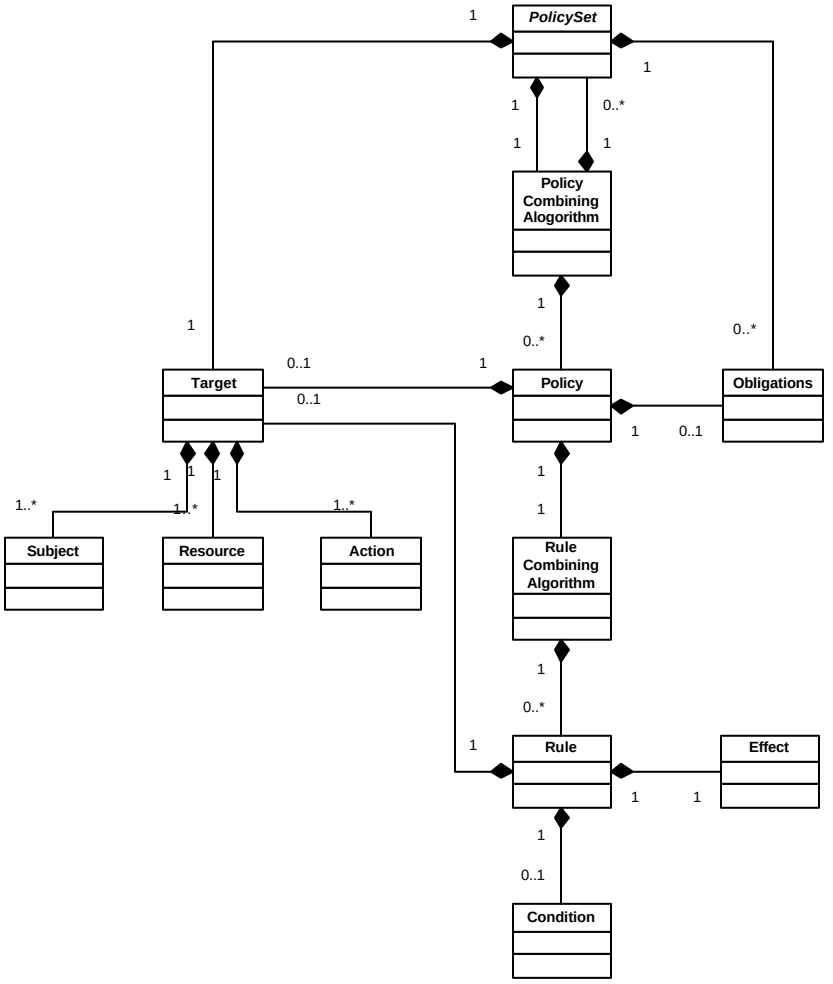

Ilustración 21. Modelo del Lenguaje de Políticas XACML (extraída de la especificación).

La regla es la unidad más elemental de una política. Para que los actores XACML puedan intercambiarse reglas deberán encapsularlas en un elemento *Policy*. Los componentes principales de un elemento *Rule* son el objetivo a quién o qué aplica la regla (elemento *Target*), el efecto que produce su ejecución (elemento *Effect*), y las condiciones que restringen su dominio de aplicación. Veamos más detenidamente cada uno de estos posibles elementos:

- ***Target* u Objetivo de una regla**. Un elemento *Target* puede definir un conjunto de recursos (elemento *Resource*), sujetos (elemento *Subject*) y acciones (elemento *Action*). Estos elementos definen la entidad objetivo sobre la que se debe aplicar la regla. Una regla se aplicará en función a si la entidad contra la que se evalúa coincide con el perfil del objetivo descrito mediante este elemento.

  Como veremos más adelante, mediante las condiciones (elemento *Condition*) aún se puede restringir más el dominio de aplicación de cierta regla. Supongamos que cierto recurso A desea ser accedido y llega una petición de acceso al PDP. Cuando éste pase a evaluar las reglas, sólo tomará aquellas cuya entidad a las que aplican (descritas mediante el elemento *Target*) coincidan con el recurso A. En el caso de que se quisiera aplicar la regla a todas las entidades de un tipo de datos en particular se podría utilizar un elemento denominado *AnySubject*, *AnyResource* o *AnyAction*, respectivamente. Por tanto, un PDP comprueba que los sujetos,

recursos y acciones identificadas en la petición del contexto XACML se encuentran presentes en el objetivo (elemento *Target*) de las reglas que emplea para evaluar la *petición de decisión*.

Una regla podría no contener un elemento *Target* en cuyo caso el objetivo al que aplica la regla es el mismo que aquel al que aplica su elemento padre *Policy*. Ciertas formas de los nombres de un sujeto, recurso y ciertos tipos de recursos se encuentran internamente estructuradas. Por ejemplo, los nombres cuya estructura procede de X.500 o del RFC 822 son formas de nombres estructuradas para los sujetos. Un ejemplo de nombre estructurado para un recurso es el nombre del camino de un sistema de ficheros UNIX o la URIs. Por su parte, un ejemplo de un recurso estructurado podría ser un documento XML. En general, el nombre de un nodo (que no sea una hoja del árbol) que se encuentre estructurado de alguna manera será una forma de nombre válida. Por ejemplo, el nombre RFC 822 “medico.com” es un nombre RFC 822 legal que identifica el conjunto de direcciones de correo hospedadas por el servidor de correo medico.com. O por ejemplo la expresión XPath/XPointer *//ctx:ContenidoRecurso/md:registro/md:paciente* es un valor XPath/XPointer legal que identifica un conjunto de nodos de un documento XML.

Debido a esta versatilidad en cuanto a la forma de nombrar a las cosas, puede surgirnos rápidamente la siguiente duda: ¿Cómo

debería ser interpretado por el PDP un nombre que identifica un conjunto de sujetos o recursos, si aparece en una política o en una petición contextual? ¿Es su intención representar explícitamente el nodo identificado por el nombre, o tienen como propósito representar el subárbol entero subordinado a ese nodo?

En el caso de los sujetos, no existe una entidad real que se corresponda con un nodo de este tipo. Por lo tanto, los nombres de este tipo se refieren siempre al conjunto de sujetos subordinados reflejados en la estructura del nombre para el nodo identificado. En consecuencia, no se deben emplear nombres de sujetos que representen nodos no hoja en funciones de igualdad y sí en funciones de "matching" como por ejemplo "urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match" y no "urn:oasis:names:tc:xacml:1.0:function:rfc822Name-equal".

En el caso de los nombres de los recursos y de los recursos mismos existen tres posibles alternativas a las que se podría referir el nodo:

- Solamente se refiere a los contenidos del nodo identificado.
- Se refiere a los contenidos del nodo identificado y a los contenidos de sus nodos hijos inmediatos.
- Se refiere a los contenidos del nodo identificado y a todos sus nodos descendientes.

Estas tres opciones están previstas y soportadas por XACML.

- ***Effect o elemento efecto de una regla.*** El efecto de la regla refleja la consecuencia definida por el autor de la regla en el caso en que ésta se evalúe a verdadero. Los dos valores permitidos como resultado de evaluar una regla son 'Permit' que significa permiso aceptado o 'Deny' que significa que el acceso ha sido denegado.

- ***Condition o elemento condición de una regla.*** Contiene una expresión "boleana" que refina el dominio de aplicabilidad de la regla. Más en concreto especializa la aplicabilidad a partir del objetivo inicial especificado por el elemento *Target*. Este elemento podría estar ausente y permitiría expresiones del tipo "sólo aplicar esta regla si el sistema se encuentra entre 10:00 y las 12:30 horas".

6.3.5 Política (elemento *Policy*)

Siguiendo el modelo que describe el flujo de los datos, podemos ver que las reglas no son elementos que sean intercambiados entre los sistemas XACML. Los elementos intercambiados son las **políticas** que contienen las reglas. Una política se compone principalmente de cuatro elementos:

- **Un elemento *Target* u objetivo**. Que permite conocer cuando debe ser aplicada las reglas contenidas en la política.
- Un elemento ***RuleCombiningAlgId*** que identifica el algoritmo de combinación de las reglas.

- ***Un conjunto de elemento Rule o reglas***. Este elemento ya ha sido descrito con anterioridad.
- **Un conjunto de obligaciones**.

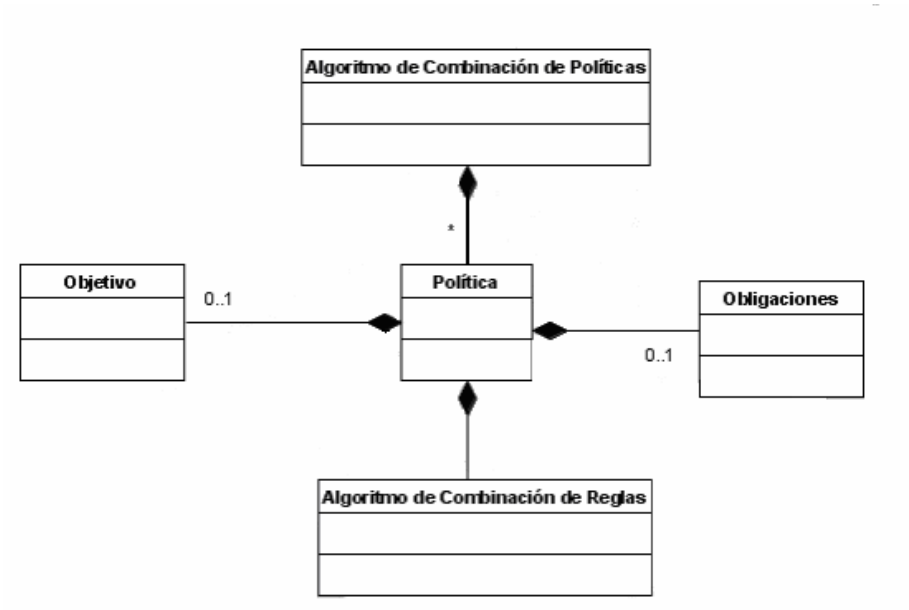

Ilustración 22. Subconjunto del modelo del lenguaje de políticas referente a la política.

Objetivo de una regla

Los elementos que pueden especificar un objetivo de aplicabilidad, mediante un elemento *Target* son *PolicySet*, *Policy* y *Rule*. Un elemento *Target* especifica los **sujetos**, **recursos** y **acciones** sobre los cuáles el conjunto de políticas, la política y las reglas respectivamente aplican.

El objetivo de una política puede ser calculado dinámicamente, aunque esto no está formalizado por la especificación XACML, a partir de los objetivos definidos en las reglas, políticas o conjunto de políticas. Este cálculo dinámico se podría llevar a cabo de dos maneras:

- El primer método consiste en que el objetivo final "neto" se compone de la **unión** de los objetivos contenidos en el elemento *PolicySet*, *Policy* y *Rule* que aplican para cierta petición. En este caso, la petición de decisión se evaluará como afirmativa si el objetivo de la regla aplica a alguno de los elementos de la unión.
- El segundo método es igual que el primero pero la operación a aplicar es la **intersección** en vez de la unión. En este caso, el objetivo será aplicable sólo a peticiones de decisión que contengan todos y cada uno de los elementos contenidos en todos los sub-objetivos de la intersección.

En los casos en los que el objetivo de una política sea declarado explícitamente por el autor de la política, cualquier regla en la política que contenga el mismo objetivo que la política podrá omitir el elemento *Target*. Es decir, los objetivos de las políticas se heredan de los elementos *PolicySet* hacia los elementos *Policy* y de éstos hacia los elementos *Rule*.

En resumen, una política se asocia con un objetivo mediante un elemento *Target*. Este elemento puede ser definido a nivel de un conjunto de políticas (elemento *PolicySet*), a nivel de una política (elemento *Policy*) o a nivel de una regla contenida en una política (elemento *Rule*). El cálculo

del objetivo sobre el que aplica cierto conjunto de políticas, políticas individuales y reglas, puede ser calculado dinámicamente mediante la unión o intersección de estos elementos que aplican a cierta petición o bien, puede ser especificado por el autor a nivel de alguno de estos tres tipos de elementos. El objetivo especificado desde un conjunto de políticas (*PolicySet*) se hereda a las políticas (elemento *Policy*) que contenga y, a su vez, a las reglas (elemento *Rule*) que la política comprenda.

Algoritmo de Combinación de Reglas

El algoritmo de combinación de las reglas define el procedimiento por el cual se combinarán los resultados de evaluar las reglas contenidas en una política. A partir del resultado obtenido por cada regla, se aplicará cierto algoritmo de combinación que generará la decisión de autorización final.

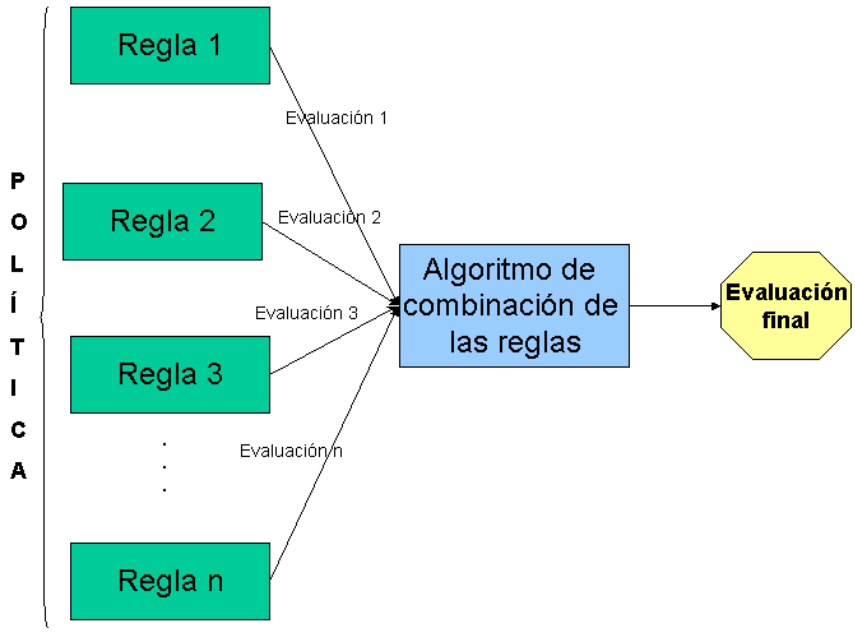

Ilustración 23. Esquema del funcionamiento de los algoritmos de combinación de las reglas.

En el apéndice de la especificación se definen un conjunto normativo de algoritmos de combinación de reglas.

Obligaciones

Este elemento se incluye dentro del elemento *Policy*. Cuando un PDP evalúa una política que contiene obligaciones, devuelve cierto conjunto de esas obligaciones al PEP en el contexto de la respuesta.

6.3.6 Conjunto de Políticas (elemento *PolicySet*)

Un conjunto de políticas se compone de cuatro componentes principales:

- Un objetivo.
- Un identificador de un algoritmo de combinación de políticas.
- Un conjunto de políticas.
- Unas obligaciones.

Los dos primeros componentes, el objetivo (elemento *Target*) y las políticas (elemento *Policy*) ya han sido descritos. A continuación comentaremos brevemente los otros dos tipos de elementos:

- ***Algoritmo de Combinación de Políticas.*** Este elemento define el procedimiento por el cual se deben combinar los resultados de evaluar cada política cuando se evalúa el conjunto de políticas. El elemento *Decision* introducido en el contexto de respuesta XACML por el PDP es el resultado de evaluar el conjunto de políticas tal y como se describe en el algoritmo de combinación de políticas.
- ***Obligaciones***. El autor de un conjunto de políticas podrá añadir obligaciones al conjunto de políticas, en suma con aquellas contenidas en los componentes de políticas y conjunto de políticas subordinados. Cuando un PDP evalúa un conjunto de políticas que contienen obligaciones, devolverá parte de estas obligaciones al PEP en el contexto de la respuesta XACML.

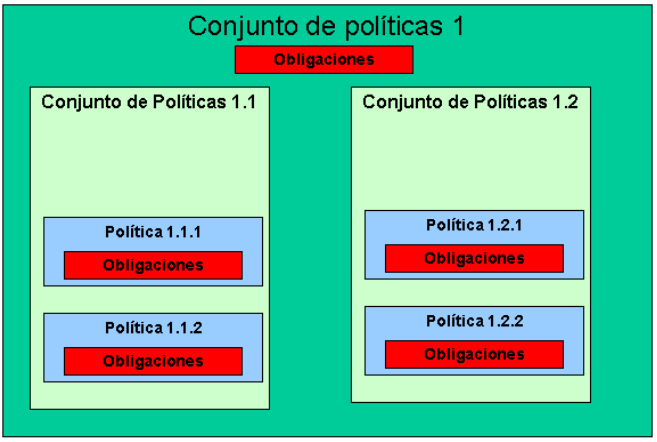

Ilustración 24. Obligaciones en los conjuntos de políticas.

6.3.7 Ejemplos

En esta sección mostraremos algunos ejemplos, extraídos de la propia especificación, con el objeto de ejemplificar todo lo explicado hasta este punto.

Ejemplo de política simple

Pensemos en cierta corporación con nombre Medi Corp. y cuyo dominio es medico.com que posee una política de control de acceso que en lenguaje natural se podría expresar de la siguiente manera:

*"Cualquier usuario con un correo electrónico en el espacio de nombres del dominio medico.com está permitido a realizar cualquier **acción** sobre cualquier **recurso**"*

Una política XACML consiste de información de cabecera, un campo de texto adicional con la descripción de la política, un objetivo, una o más reglas y conjunto de obligaciones adicionales.

La cabecera de una política XACML tiene el siguiente aspecto:

```
<Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy http://www.oasis-
open.org/tc/xacml/1.0/cs-xacml-schema-policy-01.xsd"
PolicyId="identifier:example:SimplePolicy1"
RuleCombiningAlgId="identifier:rule-combining-algorithm:deny-overrides" />
```

En este fragmento XML se indica que el documento es XML de la versión 1.0 y que el código de caracteres utilizado es UTF-8. A continuación nos encontramos con el elemento *Policy* que declara el espacio de nombres por defecto con la URI en formato URN y cuyo valor está dado por la especificación XACML. El elemento *Policy* además contiene un atributo, *xsi:schemaLocation*, indicando la ubicación del esquema XML asociado al documento, un atributo *PolicyId* que debe contener un identificador único para la política en cuestión y un atributo *RuleCombiningAlgId* que identifica el algoritmo utilizado para la combinación del resultado de la evaluación de las reglas.

El identificador de la política debe ser único en el contexto de un PDP de forma que no haya ambigüedad si una política es referenciada por otra política.

El identificador referente al algoritmo de combinación de las reglas puede ser normativo e incluido en la especificación o puede ser uno propio creado por el usuario. En este caso el algoritmo especificado es *deny-overrides* (sobreescritura de la denegación) que expresa que si alguna de las reglas evalúa a denegación entonces el resultado de la combinación será denegación. Por tanto, sólo si todas las reglas evalúan a "Permit" o permitido, la política deberá retornar permitido. Los algoritmos de combinación de reglas se encuentran ampliamente desarrollados en el apéndice C del documento principal de la especificación.

A continuación de la cabecera, en la que como hemos visto se define la URI del espacio de nombres por defecto de XACML, se asigna un identificador único a la política y se le asocia un algoritmo de combinación de las reglas, aparece un campo textual, optativo, que permite introducir información sobre la política:

```
<Description>Medi Corp access control policy</Description>
```

Tras el campo descriptivo opcional, se sitúa el elemento *Target* u objetivo que permite, como ya se ha explicado anteriormente, definir el dominio de aplicabilidad de la política. Como también ya ha sido

mencionado, un objetivo puede contener tres tipos de elementos: sujetos, recursos y acciones.

```
<Target>
  <Subjects>
        <AnySubject />
  </Subjects>
  <Resources>
        <AnyResource />
  </Resources>
  <Actions>
        <AnyAction />
  </Actions>
</Target>
```

El elemento *Target* define qué peticiones de decisión recibidas por el PDP aplicarán a esta política. Si el sujeto, recurso y acción contenido en una petición de decisión no coincide con los valores especificados en el elemento *Target*, entonces no será necesario evaluar el resto de la política. Esta sección *Target* es muy útil para crear un índice sobre un conjunto de políticas.

En este ejemplo, el fragmento permite que sea evaluada con esta política cualquier petición de decisión.

Recordemos el contenido básico de una política es “una cabecera (la declaración del espacio de nombres XACML, con el identificador único de la política, y con el algoritmo de especificación de combinación de las reglas contenidas en la política), un campo textual para introducir alguna

descripción acerca de la política, una o más reglas, etc.”. Veamos pues un ejemplo de regla muy sencillo:

```
<Rule RuleId="urn:oasis:names:tc:xacml:1.0:example:SimpleRule1"
Effect="Permit" />
```

Como vemos en el fragmento XML anterior se define una regla en una política mediante el elemento *Rule* más un par de atributos:

- RuleId**.** Contiene una URI que identifica unívocamente la regla.
- Effect. El resultado de evaluar positivamente, concretamente a “true”, la regla contra el contenido de la petición de decisión (aunque no siempre será necesario aplicar una regla contra el contenido como por ejemplo cuando la regla indique que sólo permite el acceso si la petición llega en un momento concreto en el tiempo). Este atributo puede contener dos valores: “Permit” o “Deny”. Si la regla evaluara a “False”, entonces retorna un resultado “NotApplicable” Si ocurriera un error cuando se evalúa la regla, ésta retornará “Indeterminate”. Como explicamos con anterioridad, asociado a una política, mediante el atributo *RuleCombiningAlgId*, se define el algoritmo de combinación de reglas. Este algoritmo expresa cómo deben ser combinados los resultados obtenidos de evaluar todas y cada una de las reglas que permita obtener un resultado de la evaluación global.

Dentro de un elemento regla nos podemos encontrar opcionalmente y en primer lugar con un elemento que contiene una descripción textual de la regla. Por ejemplo:

```
<Description>Cualquier sujeto que posea una dirección de correo electrónico cuyo nombre de dominio sea medico.com podrá realizar cualquier tipo de acción sobre cualquier recurso.</Description>
```

A continuación se define un elemento *Target* que representará el dominio de aplicabilidad específica para la regla.

Si el sujeto, recurso y acción en una petición de decisión no coinciden con los valores especificados en el objetivo de la regla, entonces el resto de la regla no necesitará ser evaluada, y se devolverá un valor "NotApplicable" como resultado parcial para la evaluación de la política.

```
<Subjects>
        <Subject>
                <SubjectMatch
        MatchId="urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match">
                <SubjectAttributeDesignator
        AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
        DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" />

                <AttributeValue
        DataType="urn:oasis:names:tc:xacml:1.0:data-
        type:rfc822Name">medico.com</AttributeValue>
                </SubjectMatch>
        </Subject>
</Subjects>
```

```
<Resources>
        <AnyResource />
</Resources>
<Actions>
        <AnyAction />
</Actions>
```

Como podemos ver el contenido del objetivo de esta regla es similar al de la política que la contiene salvo una pequeña diferencia. En este fragmento el sujeto que aplica se restringe al identificador del sujeto procedente desde el contexto de la petición. En este caso el elemento *SubjectMatch* especifica una función de semejanza con el atributo *MatchId*, que referencia a un atributo específico del sujeto contenido en el contexto de la petición mediante el elemento *SubjectAttributeDesignator*, y un valor literal "medico.com". La función de semejanza será utilizada para comparar el valor del atributo del sujeto con el valor del literal.

Solamente si la comparación retorna "true" se aplicará esta regla a una petición de decisión concreta. Si la comparación retorna "false", entonces esta regla devolverá un valor "NotApplicable".

En reglas más complejas, al objetivo le podría seguir un elemento *Condition* (que a su vez podría ser un conjunto de condiciones combinadas mediante los operadores lógicos AND y OR).

En resumen, hemos visto una política muy sencilla que contiene una sola regla. Dicha regla obliga a que el sujeto que realiza la petición disponga de un identificador con formato RFC 822 cuyo valor sea “medico.com”.

Ejemplo de un Contexto de Petición

En esta sección vamos a analizar un ejemplo de un hipotético contexto de petición que podría haber sido enviado a un PDP que utiliza la política descrita en el apartado anterior. En lenguaje natural, la petición de acceso que genera la petición de decisión podría ser indicada como:

> *“Bart Simpson, con correo electrónico bs@simpsons.com, desea leer su registro médico ubicado en MediCorp”*

En XACML, la información contenida en una petición de decisión se formatea en un contexto de petición que posee el siguiente aspecto:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Request xmlns="urn:oasis:names:tc:xacml:1.0:context"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context
        http://www.oasis-open.org/tc/xacml/1.0/cs-xacml-schema-context-
        01.xsd" >
```

A continuación del elemento *Request*, se incluye un elemento Subject utilizado de la misma manera que fue utilizado en la política del ejemplo anterior:

```
<Subject>
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-
id" DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name">
```

```
                <AttributeValue>bs@simpsons.com</AttributeValue>
        </Attribute>
</Subject>
```

El elemento *Subject* contiene uno o más atributos pertenecientes a la entidad que está realizando la petición de acceso. Podrían existir múltiples sujetos, y cada uno podría tener múltiples atributos. En este caso, sólo existe un sujeto, y el sujeto solamente tiene un atributo: la identidad del sujeto, expresado como un nombre de correo electrónico, es ““bs@simpsons.com”. Tras definir el sujeto que está realizando la petición de acceso aparece un elemento *Resource* que representa el recurso por el que se está realizando la petición de acceso. En nuestro ejemplo, este elemento tiene el siguiente aspecto:

```
<Resource>
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:ufs-
path" DataType="http://www.w3.org/2001/XMLSchema#anyURI">
        <AttributeValue>/medico/record/patient/BartSimpson</AttributeValue>
        </Attribute>
  </Resource>
```

Como se puede intuir del fragmento XML mostrada, el elemento *Resource*, identifica el recurso al que se está realizando la petición de acceso. La identificación se realiza mediante la definición de una serie de atributos pertenecientes al recurso para el cual el sujeto ha realizado la solicitud de acceso. Sólo puede existir un recurso por cada petición de decisión. En este caso el recurso al que se está solicitando el acceso se

identifica por un nombre estructurado al estilo de los nombres de path de los sistemas Unix:

```
/medico/record/patient/BartSimpson
```

Hasta aquí, ya tenemos una petición de decisión que contiene un sujeto, aquel que está realizando la petición, y el recurso, aquel al que se está realizando la petición de acceso. Sólo nos queda describir la acción que se desea llevar a cabo sobre el recurso. Esta acción se puede representar mediante el elemento *Action*. Este elemento contiene un elemento hijo *Attribute* que utiliza un atributo suyo, con nombre *AttributeId*, para indicar que es un atributo que representa un identificador de acción. Para ello utiliza un identificador de atributo predefinido por la especificación en su apéndice. El valor del atributo es la cadena “read” de forma que la acción solicita es de lectura sobre el recurso dado. Como ocurría con el elemento referente al recurso, sólo puede existir una acción por petición de decisión.

Un contexto de petición más complejo podría haber contenido algunos atributos no asociados con el sujeto, el recurso o la acción. Éstos podrían haber sido introducidos en un elemento *Environment* opcional a continuación del elemento *Action*.

El PDP que procese este contexto de petición localizará en su repositorio de políticas la política de seguridad. Después, comparará los elementos *Target* de la política y del contexto de petición para ver si la política

aplica al contexto de petición. En este caso, como la política contiene en su elemento *Target* los elementos *AnySubject*, *AnyResource*, *AnyAction* para los sujetos que pueden entrar, los recursos a los que se puede acceder y las acciones que se pueden llevar a cabo sobre estos últimos respectivamente, pasará a analizar las reglas contenidas en la política. Para ello obtendrá la única regla definida y de nuevo comprobará que el dominio de aplicabilidad descrito en el elemento *Target* de la regla contiene el dominio de aplicabilidad del contexto de petición. En este caso, se aceptará el recurso y la acción puesto que la regla tiene asignados los elementos *AnyResource* y *AnyAction*, sin embargo, no aceptará el sujeto puesto que el identificador especificado en el contexto de petición (contenido en el atributo *subject-id*) no contiene como nombre de dominio "medico.com" sino "simpsons.com". Por tanto, el PDP rechazará la petición de acceso realizada por el sujeto bs@simpsons.com.

Ejemplo de contexto de respuesta

Como resultado entonces, no existe una regla en esta política que devuelva como valor "Permit" para el contexto de petición visto anteriormente (aquel con el subject-id "bs@simpsons.com"). El algoritmo de combinación de reglas de la política especifica que, en este contexto, se debería devolver un resultado "NotApplicable". El contexto de respuesta tendrá el siguiente aspecto:

```
<Response xmlns="urn:oasis:names:tc:xacml:1.0:context"
xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context http://www.oasis-
open.org/tc/xacml/1.0/cs-xacml-schema-context-01.xsd" />
```

Como podemos observar en el fragmento anterior, la respuesta posee el mismo tipo de cabecera que aquel descrito para una política.
A continuación de la cabecera de la respuesta aparece en el elemento *Result* que contiene la decisión alcanzada por el PDP.

```
<Result>
  <Decision>NotApplicable</Decision>
</Result>
```

En este caso el resultado es “NotApplicable”. Recordemos que una política puede devolver “Permit”, “Deny”, “NotAppicable” o “Indeterminate”.

7. FEDERACIÓN EN SERVICIOS WEB

7. Federación en Servicios Web

Las organizaciones hoy en día poseen estructuras muy complejas formadas por numerosas unidades de negocio, tanto internas como externas, que se encuentran ubicadas en distintos puntos geográficos y además tienen establecidas entre sí todo tipo de relaciones (comerciales, operativas, etc.). La espectacular evolución que ha sufrido Internet en los últimos 10 años la convierte sin lugar dudas en el medio de conexión o en la infraestructura idónea sobre la que materializar estas distribuciones corporativas. Un ejemplo evidente de este hecho es el uso generalizado del correo electrónico como principal medio de comunicación interorganizacional o la creación de portales corporativos.

Pero no sólo vale con hacer presentes estas unidades organizativas en la Web, además, también se espera que las personas que trabajan en ellas (o sean clientes de ellas) puedan, de forma transparente, cruzar los límites de su organización para entrar en los dominios de otras corporaciones con las que su organización de origen tenga establecida una relación de confianza. Una ventaja inmediata que ofrece este escenario es que habilita un entorno Single Sign-On (SSO) en el que el usuario podrá navegar por aquellos sitios Web de confianza teniendo que identificarse solamente una vez en su sitio Web de origen.

Este concepto de federación interorganizacional basada en Internet será el tema que trataremos con profundidad en este artículo. Básicamente el artículo se divide en dos partes. En la primera parte veremos los

conceptos básicos de la federación de servicios Web, apartado 2 y 3, y en la segunda, apartado 4, haremos un breve repaso de los principales estándares y especificaciones existentes. El último apartado reflejará unas conclusiones finales.

7.1 Introducción al concepto de federación

El concepto de federación, tal y como se aplica en este contexto, consiste en la capacidad de interacción entre agentes pertenecientes a distintos dominios de confianza y, por tanto, con políticas de seguridad independientes (IBM & Microsoft, 2003) . El aspecto clave de los servicios federados es definir una forma estándar de establecer y reflejar la confianza entre las organizaciones. Obviamente estas relaciones deberán estar sujetas a un marco legal de forma que la privacidad de los usuarios y la reputación de las empresas que forman parte de una federación se sientan protegidas.

La federación tiene que ver directamente con los servicios de seguridad de autenticación y autorización.

La **federación de las identidades** Web entre dominios de confianza permite que los usuarios puedan identificarse una vez y acceder a todos ellos de manera transparente. Debemos tener en cuenta que cuando hablamos de usuarios, pueden ser tanto personas como usuarios computacionales (otros servicios Web).

El proyecto Liberty Alliance Project o el proyecto Passport de Microsoft son dos ejemplos de marcos de trabajo cuyo primer objetivo es la federación de las identidades que habiliten servicios SSO. Como veremos en el apartado 3, la federación también garantiza que las identidades locales utilizadas por un mismo usuario en distintos sitios federados se encuentren conectadas mediante técnicas de gestión de la federación (por ejemplo mediante pseudónimos).

El potencial que podemos apreciar sobre la posibilidad de que nos autentiquemos una sola vez y a partir de ese momento seamos capaces de interaccionar con servicios ubicados en otros dominios de confianza puede no ser siempre tal. De hecho, en la práctica no nos interesa tener una única identidad en la Web sino varias: una identidad cuando accedemos desde casa, otra cuando accedemos desde la oficina, etc. En realidad el gran potencial de un servicio SSO está en que permita que una entidad defina el número exacto de las identidades Web que desea tener.

La federación comienza por lo tanto a partir de la noción de identidad federada. Es decir, el solicitante o alguien en quién éste delega (un servicio Web que actúa como proveedor de identidades con la autoridad propietaria de la información de identidad de los principales de su dominio) declara una identidad y un proveedor de identidades la verifica.

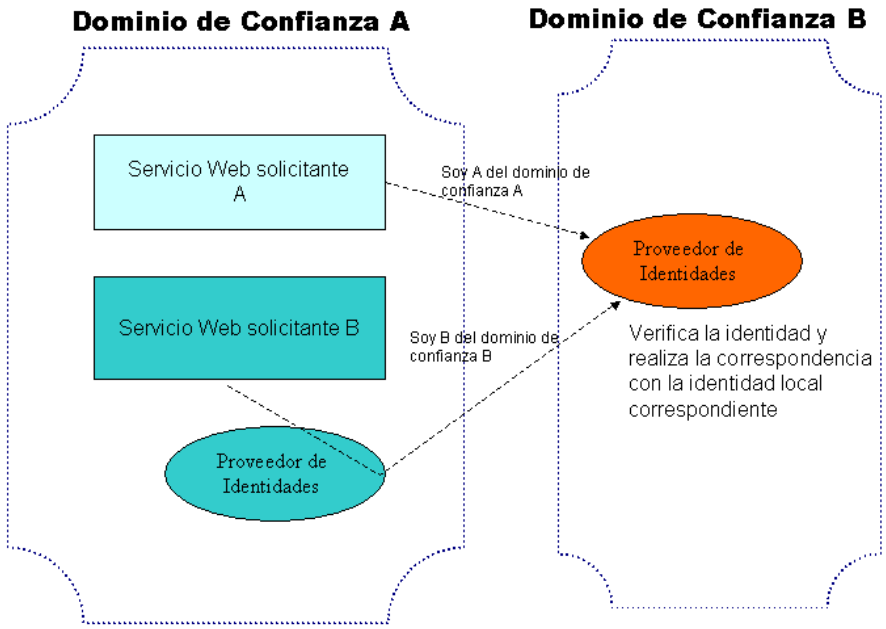

Ilustración 25. Autenticación directa (Servicio Web solicitantes A) o delegada a un proveedor de identidades (Servicio Web solicitante B) .

La **federación de la autorización** se consigue mediante la federación de los atributos de privilegio de los usuarios de forma que un servicio Web que actúa en cierto dominio de confianza pueda ejecutar sus políticas de acceso a partir de los privilegios de los usuarios pertenecientes a otros dominios de confianza federados. La arquitectura original basada en Internet que trata este problema es la definida por el proyecto Shibboleth (Internet2/MACE, 2002). Entre los estándares que permiten la portabilidad de los elementos de seguridad entre los diferentes dominios

de confianza están WS-Trust (S. Anderson, J. Bohren, T. Boubez, M. Chanliau, G. Della-Libera, B. Dixon, P. Garg, E. Gravengaard, M. Gudgin, P. Hallam-Baker et al., 2004), mediante los protocolos que permiten la emisión, renovación, validación y conversión de tokens de seguridad firmados por entidades de confianza, y SAML (Farrell et al., 2003), mediante la expresión en XML de declaraciones de autenticación, autorización y de atributos.

Otro aspecto fundamental de la federación es la **privacidad**. En este contexto la privacidad se debe considerar de manera doble: privacidad de las identidades de los usuarios y la privacidad de los atributos federados entre los dominios de confianza.

La **privacidad de la identidad** consiste en que sea imposible (idealmente) conocer la traza de la actividad llevada a cabo por un usuario (identidad) en su navegación por los sitios ubicados por los distintos dominios de confianza federados. La solución a este problema se basa en el uso de pseudónimos que impidan a los distintos servicios accedidos por un usuario vincular sus acciones.

Por su parte la **privacidad de los atributos** consiste en el desvelamiento controlado por políticas de privacidad de los atributos federados. Es decir, se comparten los atributos de los usuarios entre los dominios pero se revelan de forma controlada mediante políticas de privacidad específicas definidas por los propios usuarios.

7.2 Metas

Las metas que persigue la federación de las identidades son:

- Reducir el coste de la gestión de las identidades reduciendo el esfuerzo de duplicación de forma que cada identidad sólo debe ser conocida por una organización de confianza.
- Aprovechar el trabajo hecho los gestores de las identidades existentes dando acceso a otras partes según sea requerido y aplicando las protecciones de privacidad de la manera adecuada.
- Preservar la autonomía de las partes de forma que si una de las autoridades de identidades selecciona una tecnología de autenticación específica el resto no tenga por qué verse en la obligación de tener que utilizar también esa tecnología.
- Respetar las estructuras de confianza y los contratos existentes. El hecho de que un servicio Web acepte recibir información de identidad desde un proveedor de identidades requiere que la organización a la que pertenezca el servicio Web sólo tenga que establecer una relación de confianza con la organización a la que pertenece ese proveedor de identidades.
- Proteger la privacidad de los individuos respetando y aplicando de manera robusta las preferencias, indicadas por los usuarios, que gobiernan el uso de la información de identidad, teniendo en cuenta las reglas de privacidad regionales y gubernamentales, garantizando el consentimiento por parte de los usuario de nuevos usos e implementando fuertes mecanismos de rastreabilidad que

garanticen que las prácticas de seguridad se están llevando a cabo correctamente.

- Construir estándares abiertos que habiliten transacciones fiables y seguras para los negocios y los individuos.

7.3 Agentes

Para habilitar la federación, Microsoft e IBM introducen, en su especificación WS-Federation, los siguientes servicios:

- Proveedores de identidad. Proporcionan identidades las cuales son utilizadas para realizar la correspondencia/indexación con la información de identidad local. Es decir, proporcionan una identidad a un servicio sobre un cliente que permite al primero asociar esa identidad con una identidad local. Gracias al uso de los mecanismos de confianza (ofrecidos por ejemplo por las especificaciones WS-Trust (S. Anderson, J. Bohren, T. Boubez, M. Chanliau, G. Della-Libera, B. Dixon, P. Garg, E. Gravengaard, M. Gudgin, P. Hallam-Baker et al., 2004) o SAML (Farrell et al., 2003) y la federación
- Servicios de atributos. Proporcionan un mecanismo para poder acceder a atributos federados pertenecientes a identidades federadas. Un registro UDDI o un servicio de directorio X.500 son alternativas tecnologías para implementar un servicio de atributos específico.
- Servicios de pseudónimos. Proporcionan un mecanismo de correspondencia que pueden ser utilizados para facilitar el 'mapping' de identidades federadas de confianza de forma que se

proteja su privacidad e identidad. Un servicio de pseudónimos permitiría a un usuario conocido como foo@example.com en el dominio de confianza A (federado con un dominio de confianza B) poseer un pseudónimo en B de forma que su navegación a través de A y B no pueda ser monitorizada o identificada. En B este usuario tendrá una identidad que se corresponderá con foo@example.com. Esa identidad se corresponderá a su vez con un pseudónimo de forma que sea imposible que fue foo@example.com del dominio de confianza A el que utilizó ciertos servicios del dominio de confianza B.

7.4 Procesos de federación

En este apartado vamos a recorrer los distintos conceptos relacionados con la federación mediante un escenario de ejemplo.

Imaginemos dos típicos portales ficticios ejemplo-vuelos.com y ejemplo-coches.com que se dedican a vender billetes de avión y a realizar alquileres de vehículos a sus usuarios registrados, respectivamente. Un buen día deciden establecer una relación comercial de forma que los usuarios que acceden desde alguno de los dos sitios al otro recibirán una rebaja de un x % en la compra o reserva de sus productos. Además, se desea que si un usuario se autentica correctamente en alguno de los dos sitios no tenga por qué volver a autenticarse en el otro.

Para ello, en primer lugar ambos sitios Web deberán **definir el marco legal** bajo el que colaborarán de forma que se sientan respaldados

legalmente. Este marco legal será la protección respaldo tangible para la relación de confianza que establezcan.

Como primera medida ven la necesidad de **federar las identidades** de sus usuarios. Para ello, introducen en sus portales respectivos una nueva opción que invita a los usuarios, una vez autenticados, a federar su identidad con el otro portal. Si el usuario acepta esta propuesta, ambos sitios Web deberán, de alguna manera comunicarse las identidades locales de ese usuario de forma que puedan conocer con certeza qué usuario es en cada momento. Imaginemos que Juan tiene una única cuenta en ejemplo-vuelos.com, bajo la identidad JuanVuelos y que un día decide registrarse en el sitio ejemplo-coches.com bajo la identidad de JuanCoches. La siguiente vez que se autentica en ejemplo-vuelos.com se le invita a federar esa identidad con el sitio ejemplo-coches.com y, gracias a la llamativa rebaja, acepta sin dudar. En ese momento, entre ambos sitios Web deberán llevar a cabo algún proceso que les permita intercambiarse las identidades locales de Juan. Por ejemplo (ver figura 1), el portal ejemplo-vuelos.com podría redirigir el navegador de Juan al sitio ejemplo-coches.com incluyendo en la URL un testigo de petición de federación de la identidad de ese usuario redirigido (2). En ese momento ejemplo-coches.com autentica al usuario (2), es decir conoce que su identidad es JuanCoches, y procede a federar esa identidad con el identificador JuanVuelos incluido en el testigo (3). Así mismo redirigirá al usuario de nuevo a ejemplo-vuelos.com incluyendo en la URL un testigo de respuesta a la petición de federación que incluya la identidad de Juan en ejemplo-coches.com (4). En ese momento se completará la federación de las identidades locales de Juan en ambos dominios (5).

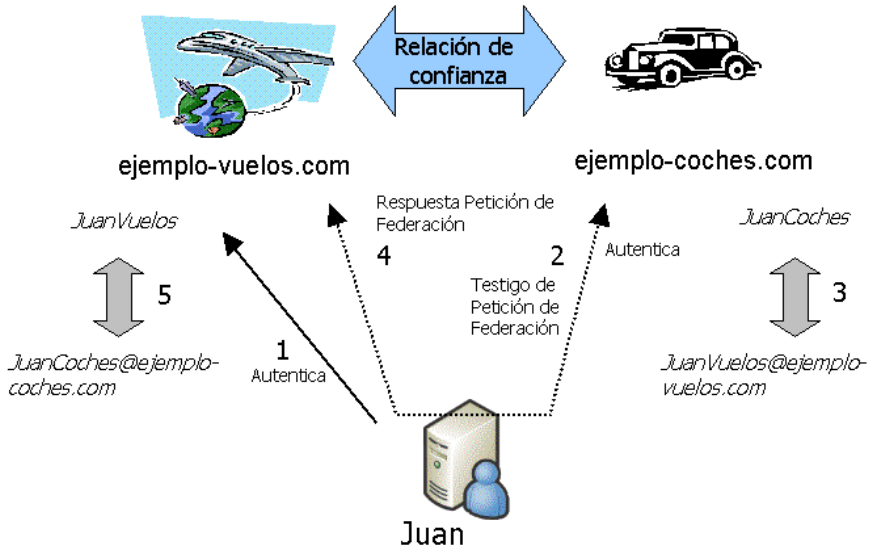

Ilustración 26. . Federación de identidades locales de un mismo usuario ubicadas en dominios de confianza federados.

Una vez Juan ha federado sus identidades, podrá utilizar el **servicio SSO** (ver figura 2). La siguiente vez que Juan se autentique en ejemplo-vuelos.com (1), realice su reserva de avión, y desee dirigirse a ejemplo-coches.com para realizar la reserva del coche, se le redirigirá adjuntándole algún tipo de testigo que evidencie que el sitio ejemplo-vuelos.com ya le autenticó (2). Ese testigo puede simplemente contener un identificador del testigo (un flujo de octetos lo suficientemente grande

como para que no sea posible su adivinación) y otro identificador del sitio ejemplo-vuelos.com. Cuando ese testigo le llegue a ejemplo-coches.com, sabrá que procede de ejemplo-vuelos.com y entonces le realizará una petición, tomando el rol de servicio Web consumidor, solicitándole una confirmación de que, efectivamente, autenticó al usuario que le presenta ese testigo (3). El servicio Web ejemplo-vuelos.com devolverá una respuesta afirmativa en la que incluirá información acerca del proceso de autenticación que llevó a cabo para ese usuario incluyendo su nombre de usuario, JuanCoches, en el dominio del solicitante. En ese momento, ejemplo-coches.com sabrá que fue efectivamente JuanCoches quién fue autenticado y asociará todas las acciones que lleve a cabo con esa identidad. Este esquema SSO es un ejemplo de entre los muchos perfiles SSO que definen las especificaciones que veremos en el apartado 4.

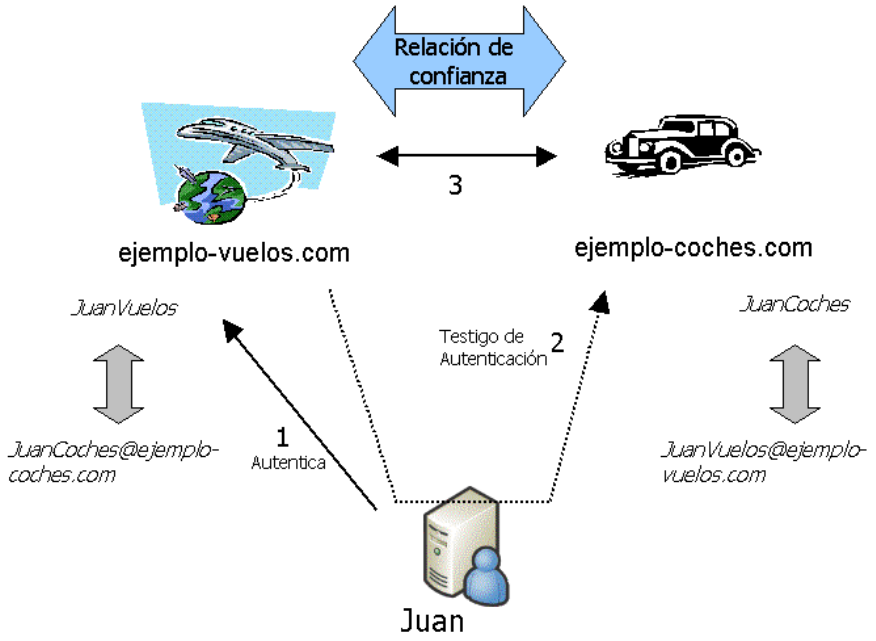

Ilustración 27. Esquema general de autenticación SSO.

Fenomenal, pero todo esto parece poco realista, ¿cómo podemos pensar que una organización va a revelar su directorio de usuarios a otra organización? ¿Cómo va un servicio de banca en Internet a proporcionar las identidades de sus usuarios a un servicio de compra de acciones en bolsa sin la consiguiente preocupación sobre la privacidad de los usuarios?

Este problema se puede resolver si durante el proceso de federación, los dominios no se intercambian las identidades locales sino unos **pseudónimos** que estén asociados con las identidades locales de Juan en

cada uno de los dominios. Durante el proceso de federación de las identidades, ejemplo-coches.com puede enviar un pseudónimo, por ejemplo *asdf* a ejemplo-vuelos.com, y después de que ejemplo-vuelos.com autentique al usuario le asocia el pseudónimo *poiuy* que comunica al servicio ejemplo-coches.com. De esta forma, ejemplo-vuelos.com sabe que cuando JuanVuelos deba ser redirigido a ejemplo-coches.com se deberá enviar la identidad *asdf* mientras que cuando ejemplo-coches.com haga la petición del testigo que confirma que el usuario fue autenticado, enviará el pseudónimo *poiuy*. Normalmente, y con el objetivo de mantener el anonimato, las partes cooperantes querrán, cada cierto tiempo renovar estos pseudónimos y, por lo tanto, deberán ser capaces de renovar los pseudónimos mediante algún **protocolo de federación de renovación de pseudónimos** sin deshacer la federación existente.

Si existiera un nuevo sitio Web que quisiera establecer una nueva relación de confianza con ejemplo-vuelos.com se federarán las identidades otorgando otro par de pseudónimos distintos de forma que entre en el nuevo sitio Web y ejemplo-coches.com será imposible relacionar las acciones que Juan ha realizado.

Una vez Juan haya completado sus actividades en el portal ejemplo-vuelos.com (o simplemente se termina su sesión por superar el tiempo de inactividad) podría señalizar su intención de terminar la sesión de forma que se deberá llevar a cabo un proceso de **Single Log-Out (SLO)**, también conocido como Single Sign-Out (ver figura 3). En este caso, el

sitio Web ejemplo-vuelos.com deberá comunicar a ejemplo-coches.com que se dio por finalizada la sesión del usuario (3). A partir de ese momento ambos sitios limpiarán sus cachés locales que contienen la información sobre el proceso de autenticación SSO que llevó a cabo el usuario Juan de forma que la siguiente vez que desee volver a utilizar los servicios de la federación tenga que volver a autenticarse (2) y (4). Debemos tener en cuenta que cuando un usuario realiza logout podría pretender hacerlo en el contexto de la federación completa o tan sólo del sitio Web en el que se encuentra, por ello, normalmente el sitio Web dispondrá de dos opciones para realizar el logout de forma que se contemplen ambas posibilidades.

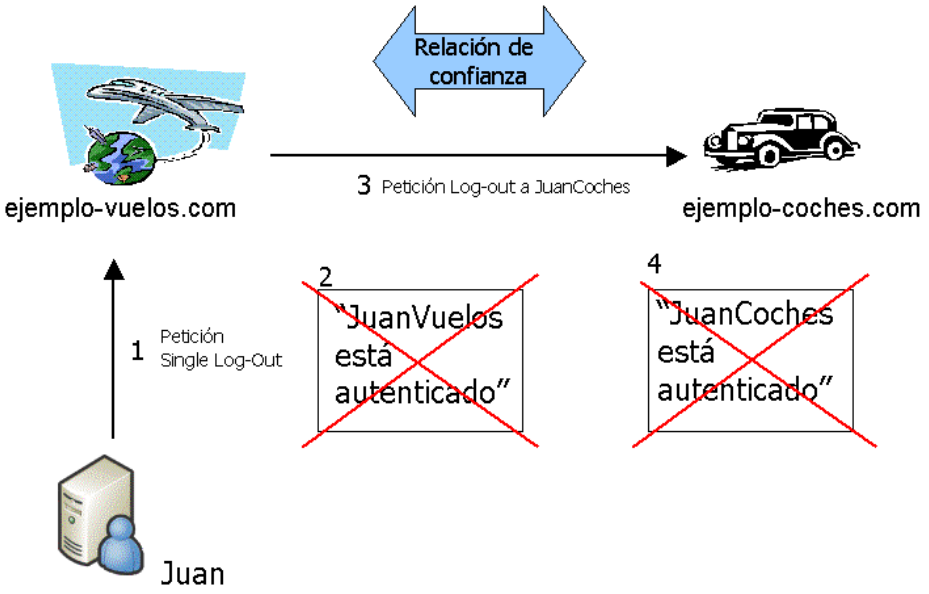

Ilustración 28. Pasos seguidos durante el proceso de Single Sign-Out.

Surge otro problema más, la compañía propietaria del sitio Web ejemplo-coches.com ha sido comprada por otra empresa que impone, como una de las cláusulas para efectuar la adquisición que el sitio Web ejemplo-coches.com deshaga la federación con el sitio ejemplo-vuelos.com. Se deberá llevar a cabo un **proceso de terminación indefinida de la federación** de las identidades de forma que ejemplo-vuelos.com no disponga de ningún pseudónimo para Juan para el dominio ejemplo-coches.com y viceversa. Para ello, ejemplo-coches.com deberá llevar a cabo algún tipo de protocolo con el servicio Web de ejemplo-vuelos.com

que se encargue de la terminación de forma que se desvinculen las identidades Web de los usuarios de ambos dominios.

Existen más posibilidades dentro del marco de la federación como es la **federación de los atributos** de los usuarios. Esta posibilidad podría mejorar cualitativamente la experiencia de los usuarios de la Web. Imaginemos que cuando Juan se registró en su momento en ejemplo-vuelos.com, tuvo que introducir su dirección postal. Imaginemos también que durante el proceso de registro que Juan efectúo en ejemplo-coches.com no le fue preguntado tal atributo. Durante el proceso de federación de las identidades, y después de que Juan se autentique en ejemplo-vuelos.com, le es preguntado si acepta compartir su dirección postal con el sitio ejemplo-coches.com a lo que responde afirmativamente.

Cierto día Juan se autentica en ejemplo-vuelos.com y, tras solicitar su deseo de alquilar un coche, es redirigido a la Web ejemplo-coches.com, llevándose a cabo el proceso SSO. Una vez alquila su coche, el sistema ejemplo-coches.com realiza una petición, como servicio Web cliente, al servicio Web que sirve los atributos de los usuarios de ejemplo-vuelos.com, solicitándole la dirección postal de Juan. Esta petición se realizará en nombre del pseudónimo con el que ejemplo-coches.com sabe que se tiene que referir a Juan en el contexto ejemplo-vuelos.com. El servicio de atributos de ejemplo-vuelos.com responde con dicho atributo y el sitio Web ejemplo-coches.com, al ver que el usuario pertenece a

cierta zona geográfica, aplica una tarifa aún más barata que es comunicada al usuario a través de la Web.

El uso de los atributos puede ser tan trivial como el mencionado o podría ser utilizado por el **servicio de autorización** de ejemplo-coches.com. Imaginemos que Juan es menor de edad y que esta información, que él introdujo en ejemplo-vuelos.com, fue también federada en el momento de federar la dirección postal. Cuando Juan es redirigido a ejemplo-coches.com, el servicio de autorización de ejemplo-coches.com podría solicitar el atributo que representa la edad de Juan al servicio de atributos de ejemplo-vuelos.com y, al ver que es menor de edad, no permitirle realizar el alquiler del coche.

Por supuesto, para que esto sea válido, la fuente de los atributos de cierta identidad deberá haber contrastado su autenticidad. Por ejemplo, durante el proceso de registro de Juan en el sitio ejemplo-vuelos.com, y como un paso más, tuvo que personarse con su NIF en una delegación de ejemplo-vuelos.com.

Otro aspecto más sobre la federación de los atributos es que los dominios de confianza federados deberán acordar una nomenclatura y semántica común para los nombres de los atributos.

Por último señalar que para que todos estos protocolos se puedan seguir, cada sitio Web cooperante deberá distribuir cierta **meta-información de federación** a los otros sitios Web de la federación como por ejemplo: la

ubicación donde solicitar la federación de la identidad, enviar las peticiones de los testigos que confirman la autenticación de cierto usuario, o enviar los mensajes de SLO. Esta meta-información puede ser intercambiada *off-line* entre los participantes, como por ejemplo durante la creación del marco legal, o bien puede ser descubierta de manera dinámica utilizando los mecanismos definidos en ciertas especificaciones como por ejemplo XACML (Anderson et al.), WS-Policy (Bajaj et al., 2004) o WS-MetadaExchange (Ballinger et al., 2004).

En este apartado hemos visto un caso práctico (que a decir verdad es poco realista aunque cumple con sus fines didácticos) en el que se mencionan los conceptos fundamentales de la federación entre sitios Web.

A continuación daremos un repaso a las especificaciones y los estándares que actualmente se relacionan con este tema.

7.5 Estándares y especificaciones

En este apartado revisaremos las soluciones actuales vinculadas con el problema de la federación de dominios de confianza.

7.5.1 Proyecto Shibboleth

El proyecto Shibboleth (Internet2/MACE, 2002) es un proyecto de Internet2/MACE cuyo capital intelectual y financiero es proporcionado por IBM. Su propósito es el desarrollo de arquitecturas, marcos de trabajo, y tecnologías prácticas que permitan la compartición

institucional de recursos que se encuentran sujetos a controles de acceso. Shibboleth define una arquitectura que permite el intercambio seguro de los atributos de privilegios (información de autorización de los sujetos) que puede ser utilizado cuando se realizan decisiones de control de acceso.

Shibboleth trata de simplificar el proceso de administración de las identidades federadas entre instituciones cooperantes. En el marco de la administración federada un proveedor de un recurso relega la administración de las identidades y los atributos de los usuarios al sitio original al que pertenecen. El proveedor de un recurso de apoya en el sitio de origen para obtener los atributos de los solicitantes y, de esta forma, poder realizar la decisiones de control de acceso oportunas.

Shibboleth es un sistema que permite la transferencia segura de atributos de un usuario desde su sitio de origen hasta el sitio al que pertenece le proveedor de los recursos. Shibboleth asume que el solicitante de los recursos está navegando utilizando un navegador común y los recursos están accesibles mediante tecnologías de Internet estándar.

Shibboleth tiene en cuenta el aspecto de la privacidad y permite que el usuario puede escoger qué información (atributos) pueden ser revelados y a qué sitios. Ya que Shibboleth tiene en cuenta la privacidad, uno de sus componentes fundamentales es la entidad que libera los atributos de los usuarios. Esta entidad, conocida en la arquitectura Shibboleth, como la Autoridad de Atributos (AA) se encarga de almacenar y liberar, bajo las

políticas de privacidad adecuadas, los atributos de los usuarios de un dominio de confianza.

Shibboleth no define por completo la implementación de un AA. Sin embargo, sí especifica una manera para los AA estructuren las políticas de liberación de los atributos. Cada dominio de confianza deberá proporcionar un servicio a través de su AA que permita a los usuarios definir las políticas de privacidad a aplicar sobre sus atributos.
A continuación presentaremos la arquitectura definida por el proyecto Shibboleth.

La siguiente figura muestra los componentes principales de la arquitectura y sus relaciones durante el proceso de autenticación de cierto usuario:

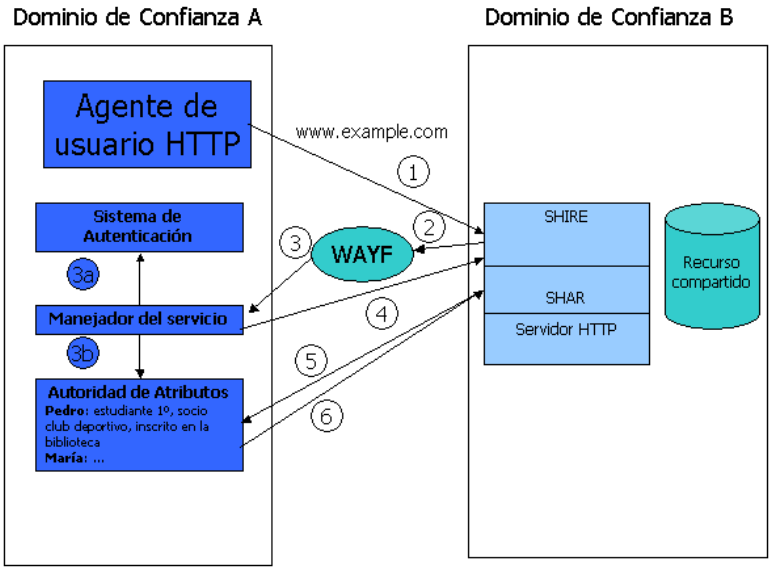

Ilustración 29. Arquitectura general del sistema Shibboleth.

En un primer momento cierto usuario que pertenece a un dominio de confianza A introduce en su navegador la URL www.example.com correspondiente a una organización ajena a él pero con la que su empresa posee una relación de confianza. En el lado del servicio conforme con Shibboleth que hospeda el sitio Web en cuestión podemos observar tres partes fundamentales: SHIRE (Shibboleth Indexical Reference Establisher), SHAR (Shibboleth Attribute Requester) y el servicio HTTP.

La petición realizada por el agente del usuario llega en primer término al componente SHIRE. Este componente se encargará de validar un

manejador asociado con el usuario que éste deberá obtener de su servicio de autenticación local.

El componente SHIRE redirige al usuario al servicio WAYF (“Where Are You From”) pasándole como argumentos la URL objetivo a la que el usuario deseaba acceder así como una URL (relativa al componente SHIRE) en la que el usuario deberá presentar un manejador indicativo de que fue correctamente autenticado en su dominio de origen. Ese manejador será evaluado por el componente SHIRE que determinará si es o no válido. El componente WAYF permitirá al usuario la introducción del sitio Web del que procede. Por ejemplo, el usuario que está intentando acceder al sitio www.example.com podría estar accediendo desde www.institutionABC.example.com. El WAYF, a partir del nombre del sitio de procedencia deberá ser capaz de obtener la dirección del manejador del servicio ante el cual se debe autenticar el usuario. Con esta información redirige el agente del usuario a dicho manejador del servicio que procederá a autenticarle. Una vez autenticado creará el token que evidencia la autenticación, y lo enviará junto con la dirección del servicio de atributos origen a la URL en la que el componente SHIRE debe realizar la validación del token de autenticación. Si el componente SHIRE valida el token de autenticación habrá identificado correctamente al usuario. A continuación el gestor del recurso deberá aplicar las reglas de acceso oportunas sobre la petición de forma que pueda evaluar si permite o no el acceso al recurso que protege. Para realizar esta evaluación necesitará con gran probabilidad atributos asociados con el usuario. Estos atributos son recuperados gracias a la información de la

dirección del servicio de atributos recibido cuando se realizó la autenticación y será llevado a cabo por el componente SHAR. Este componente actuará a favor del gestor del recurso proporcionándole toda la información necesaria y posible (que cumpla con las políticas de privacidad establecidas entre ambas organizaciones). A partir de esta información el gestor del recursos evalúa su política de control acceso y determina si permite o no el acceso al recurso.

7.5.2 Passport de .NET

La tecnología Passport de Microsoft pretende ofrecer una serie de funciones sencillas a partir de una red distribuida y uniforme. La función más importante es la autenticación. Esta funcionalidad permite que cualquier usuario que posea una credencial Passport pueda acceder a un conjunto de sitios Web que son conformes con el esquema Passport. Es decir, un usuario se autentica ante el sistema Passport y obtiene unas credenciales en forma de cookie que le permiten navegar por diversos sitios Web sin la necesidad de tener que volver a autenticarse (es decir, ofrece una solución SSO). Para que un sitio Web permita credenciales Passport primero deberá establecer e intercambiar una clave secreta con el sistema Passport. Esta clave secreta, intercambiada off-line mediante un anexo en un correo electrónico seguro utilizando cifrado de clave pública, permitirá que la cookie que se otorgue a un usuario cuando se autentique en el sistema Passport se encuentre cifrada.

7.5.3 WS-Federation

En la figura 30 se presentaba la pila de especificaciones de seguridad definida por el grupo de trabajo “espontáneo” formado por, entre otras, empresas como Microsoft, Verisign o IBM.

De esta pila de protocolos la única especificación que ya ha pasado por un organismo de estandarización, en este caso OASIS, es WS-Security. El resto se encuentran en estado de borrador. En la parte más alta de la pila nos encontramos con la especificación WS-Federation (Bajaj et al., 2003). Esta especificación trata los aspectos señalados en el apartado 3 relativos con la federación aunque todavía se encuentra en una fase muy inicial de su desarrollo y, por tanto, no posee todavía carácter normativo (aunque se puede tomar como una declaración de intenciones del trabajo que se desarrollará en sus futuras versiones). Esta especificación se acompaña de dos perfiles que aplican los mecanismos que define en dos contextos: uno en el que los clientes de la federación son navegadores HTTP (denominados clientes pasivos) y otro en el que los clientes son servicios Web (clientes activos). WS-Federation define pues un modelo para construir, a partir de las especificaciones WS-* subyacentes, los siguientes mecanismos:

- Single Sign-On.
- Single Sign-Out.
- Intermediación de la confianza entre dominios.
- Servicio de atributos.
- Servicio de pseudónimos.

Como protocolo para realizar las comunicaciones entre los participantes de la federación se emplea aquel definido en la especificación WS-Trust.

7.5.4 Liberty Alliance Project

Este proyecto está compuesto por diversas organizaciones procedentes de diferentes sectores de la industria y su principal propósito es definir un marco general para la federación de las identidades en la Web. Tal y como se indica en el documento (LibertyAllianceProject, 2003) el proyecto Liberty está desarrollando una serie de especificaciones que "habilitan la gestión de las identidades de red federadas". Su volumen de entregables es muy vasto, porque tal y como ellos mismos declaran su objetivo es "ambicioso" y necesita tener varios frentes abiertos al mismo tiempo. En terminología Liberty cualquier entidad (personal o computacional) que posee una identidad Web se denomina principal, mientras que el vínculo de confianza establecido entre los distintos proveedores de identidades y servicios se denomina círculo de confianza. Con el objeto de organizarse dividen su esfuerzo en tres áreas:

Marco de Trabajo de Federación de las Identidades Liberty (ID-FF). Este marco de trabajo ofrece un camino viable para implementar soluciones SSO mediante identidades federadas. Para ello define una serie de perfiles que abarcan las siguientes funcionalidades:

- SSO y Federación. Estos dos perfiles permiten que dos servicios federen sus identidades y puedan proporcionar SSO.
- Registro de nombres. En el caso de Liberty el protocolo de federación determina que quién inicialmente asigna un

pseudónimo a la identidad sea el proveedor de identidades de forma que el proveedor del servicio se ve sujeto a utilizar este pseudónimo. Este perfil permite al proveedor de servicios solicitar a un proveedor de identidades de su círculo de confianza realizar un cambio de pseudónimo para cierta identidad federada de forma que se ajuste con la política aplicada por el proveedor de servicios en lo relativo a la asignación de identificadores de usuarios federados.

- Perfil de notificación de terminación de la federación. Este perfil define el protocolo que deben seguir las partes que forman parte del círculo de confianza que les permite finalizar de manera indefinida la federación sus identidades.
- Single Log-Out. Este perfil define el protocolo que se debe seguir por los proveedores de identidades y servicios a la hora de realizar Single Log-Out.
- Descubrimiento de un Proveedor de Identidad. Este perfil permite a un proveedor de servicios descubrir cuáles son los proveedores de identidades que un principal podría estar utilizando.
- Correlación de Identificadores de Nombres. Este perfil ofrece la posibilidad a un proveedor de servicios de obtener un identificador de nombre SAML con el cual se puede referir acerca de cierto principal de cara a una autoridad SAML determinada. Existe un perfil que además permite cifrar el valor de este identificador de nombre de forma que pueda

cruzar sitios de terceras partes sin que estos puedan reconocerlo.

Marco de Trabajo de Servicios de Identidad Liberty (ID-WSF).

Define un marco de trabajo para crear, descubrir, y utilizar servicios de identidad. Básicamente se define un modelo conceptual que ofrece la terminología más importante para estos servicios de identidad. Un ejemplo de servicio proveedor de identidades que define es el servicio 'Discovery Service'.

Especificaciones de Interfaces de Servicios de Identidad Liberty (ID-SIS)

Este módulo define un conjunto de especificaciones que permite la creación de servicios interoperables basados en ID-WSF. Estos servicios, como por ejemplo servicios de agendas, de contactos, o de alertas, son interoperables gracias a la implementación que cada uno de ellos realiza de los protocolos Liberty.

7.6 Conclusiones

Hoy por hoy la iniciativa con más solidez y experiencia en este campo es el Liberty Alliance Project que ya dispone de multitud de productos en el mercado basados en sus especificaciones.

El proyecto Shibboleth también representa una alternativa consistente y estándar (ya que está basada en los estándares SAML y XACML de OASIS) si se desea realizar la federación de organizaciones en la que los

agentes participantes son humanos que emplean su navegador Web favorito.

Como vemos el tema de la federación Web está siendo objeto de un gran estudio y desarrollo por parte de los grandes de la industria tecnológica. Tal y como se puede apreciar tras leer el apartado 4, parece evidente que será necesario realizar un esfuerzo en paralelo que ofrezca soluciones de compatibilidad y convergencia entre las distintas soluciones ya que, como hemos visto, se encuentran hoy por hoy claramente solapadas en la aproximación que realizan sobre muchos de los aspectos que componen el concepto de federación.

Diccionario

INGLÉS	ESPAÑOL
Action	Acción
Address	Dirección
Advice	Advertencia
Algorithm	Algoritmo
Anonymity	Anonimato
Application	Aplicación
Architecture	Arquitectura
Artifact	Artefacto
Assertion	Afirmación
Assetion handle	Manejador de una afirmación
Attribute	Atributo
Attribute namespace	Espacio de nombres de un atributo
Attribute Requester	Solicitante de un atributo
Attribute selector	Selector de un atributo
Attribute statement	Sentencia de un atributo
Attribute value	Valor de un atributo
Audience restriction condition	Condición de restricción de la audiencia
Authentication	Autenticación
Authentication instant	Instante de autenticación
Authentication method	Método de autenticación
Authentication statement	Sentencia de autenticación
Authority binding	Vínculo de autoridad
Authority kind	Tipo de autoridad
Authorization	Autorización
Authorization decision statement	Sentencia de decisión de autorización
Bag	Bolsa
Binary	Binario
Binary security token	Testigo de seguridad binario
Body	Cuerpo
Canonicalization	Normalización
Carried key name	Nombre de la clave transportada
Certificate	Certificado
Cipher data	Datos cifrados

Cipher value	Valor del cifrado
Code	Código
Combining Algorithms	Algoritmos de combinación
Committe	Comité
Complex	Complejo
Complex content	Contenido complejo
Complex type	Tipo complejo
Component	Componente
Conditions	Condiciones
Confirmation method	Método de confirmación
Context	Contexto
Cover Pages	Página de cobertura
Created	Creado
Credit card info	Información de la tarjeta de crédito
Data reference	Referencia a los datos
Decoding	Decodificación
Denial of service	Denegación del servicio
Deny	Denegar
Description	Descripción
Digest	Digerir
Digital Signature	Firma digital
Directory Access Protocol	Protocolo de acceso a un directorio
Effect	Efecto
Element	Elemento
Embedded	Empotrado
Encoding	Codificación
Encoding type	Tipo de codificación
Encrypted data	Datos cifrados
Encrypted key	Clave cifrada
Encryption	Ciframiento
Envelope	Sobre
Environment	Entorno
Evidence	Evidencia
Example	Ejemplo
Expiration	Expiración
Expires	Expira
Extension	Extensiones
failed authentication	Autenticación fallida
Failed check	Verificación fallida
Federation	Federación
Framework	Marco de trabajo

Handshake	Darse la mano
Hash	Resumen
Header	Cabecera
Indeterminate	Indeterminado
Indexical Reference Establisher	Establecedor de Referencia Indéxica
Invalid security token	Testigo de seguridad no válido
Issue instant	Instante de emisión
Issuer	Emisor
Key binding	Vinculación de clave
key identifier	Identificador de clave
Key info	Información de clave
Key Management System	Sistema de gestión de claves
Key name	Nombre de la clave transportada
Key reference	Referencia a clave
Key registration service	Servicio de registro de claves
Location	Ubicación
Matching	Semejanza
Message	Mensaje
Message expired	Mensaje expirado
Must undestand	Debe entender
Name	Nombre de la clave transportada
Name identifier	Identificador de nombre
Namespace	Espacio de nombres de un atributo
Number	Número
Octet	Octeto
Online Certificate Status Protocol	Protocolo del Estado de los Certificador En Línea
Open Systems Interconnected	Sistemas Abiertos Interconectados
Passport	Pasaporte
Password	Palabra de paso
Patent policy	Políticas de patentes
Path	Camino o ruta
Payment info	Información del pago
Permit	Permitir
Policy	Política
Policy Decision Point	Punto de Decisión de las Políticas
Policy Enforcement Point	Punto de Aplicación de las Políticas
Policy Information Point	Punto de Información de las Políticas
Policy Retrieval Point	Punto de Recuperación de las Políticas
Policy set	Conjunto de políticas
Privacy	Privacidad

Processing	Procesamiento
Profile	Perfil
Properties	Propiedades
Protocol	Protocolo
Prototype key binding	Vinculación de un prototipo de clave
Pseudonymity	Pseudo-anonimato
Public Key Infrastructure	Infraestructura de Clave Pública
Quality Assurante	Garantía de Calidad
Quality of Protection	Calidad de la Protección
Recipient	Receptor
Reference	Referencia
Reference list	Lista de referencias
Reliability	Fiabilidad
remaining artifact	Artefacto restante
Remote Authentication Dial-in User Service	Servicio de Conexión Telefónico Remoto de Usuarios
Request	Petición
Resource	Recurso
Response	Respuesta
Retrieval method	Método de recuperación
Role	Rol
Routing	Enrutamiento
Rule	Regla
Schema	Esquema
Security	Seguridad
Security token reference	Referencia a un testigo de seguridad
Security token unavailable	Testigo de seguridad no disponible
Sender	Emisor
Sequence	Secuencia
Service	Servicio
Signature	Firma digital
Sniffing	Rastreamiento
Spoofing	Engaño
'Standard Generalized Markup Language	
Statement	Sentencia
Status	Estado
Status code	Código de estado
Status detail	Detalle del estado
Status message	Mensaje de estado
Stolen artifact	Artefacto robado
Stream	Flujo

Subject	Sujeto
Subject confirmation	Confirmación del sujeto
Subject statement	Sentencia sobre un sujeto
Syntax	Sintaxis
Target	Objetivo
Template	Plantilla
Timestamp	Sello o marca de tiempo
Token	Testigo
Transform	Transformar
Trust	Confianza
Unsupported algorithm	Algoritmo no soportado
Unsupported security token	Testigo de seguridad no soportado
Unverified key binding	Vinculación de clave no verificada
Usage	Uso
Username	Nombre de usuario
Username token	Testigo de nombre de usuario
Web browser	Navegador Web
Where Are You From	De dónde procedes?
Working Group	Grupo de Trabajo

Acrónimos

CA	Certification Authority
CERN	European Organization or Nuclear Research
CGM Open	Computer Graphics Metafile Open
CRC	Código de Redundancia Cíclico
CRL	Certificate Revocation List
DARPA	Defense Advanced Research Projects Agency
DoS	Denial-of-Service
e-business	Electronic Business
ebXML	Electronic Business XML
ERCIM	European Research Consortium in Informatics and Mathematics
HTTP	Hypertext Transport Protocol
IETF	Internet Engineering Task Force
INRIA	Institut National de Recherche en Informatique et Automatique
IPSec	IP Security
ISO	International Organization for Standardization
KEIO	Keio University of Japan
LCS	Laboratory for Computer Science
LDAP	Lightweight Directory Access Protocol
MIT	Massachussets Institute Technology
OASIS	Organization for the Advancement of Structured Information Standards
OCSP	Online Certificate Status Protocol
P3P	Platform for Privacy Preferences
PKCS	Public Key Cryptography Standard
PKI	Public Key Infrastructure
RBAC	Role-based Access control
RFC	Request For Comments
S/MIME	Secure Multipurpose Internet Mail Extensions

SAML	Secure Assertions Markup Language
SGML	Standard Generalized Markup Language
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SSL	Secure Socket Layer
SSO	Single Sign-On
TLS	Transport Layer Security
UDDI	Universal Description Discovery Integration
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTC	Universal Time Conversion
UTF-8	Unicode Transformation Format
VPN	Virtual Private Network
W3C	World Wide Web Consortium
WSDL	Web Services Description Language
WS-I	Web Services Interoperability Organization
WSRA	Web Services Reference Architecture
XACML	Extensible Authorization Markup Language
XKMS	XML Key Management System
XML	Extensible Markup Language
XPath	XML Path
XSLT	XML Stylesheet Language Transformation

Referencias

Adams, C., & Farrell, S. (1999). *Internet X.509 Public Key Infrastructure Certificate Management Protocols*, from http://www.cis.ohio-state.edu/htbin/rfc/rfc2510.html

Anderson, A., Parducci, B., Adams, C., Flinn, D., Brose, G., Lockhart, H., Beznosov, K., Kudo, M., Humenn, P., Godik, S., Andersen, S., Crocker, S., & Moses, T. (2003). *eXtensible Access Control Markup Language (XACML) Version 1.0*, from http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

Anderson, A., Proctor, S., & Godik, S. (2004). *OASIS XACML profile for Web-services*, 2004, from http://xml.coverpages.org/WSPL-draft-xacmlV04-1.pdf

Anderson, S., Bohren, J., Boubez, T., Chanliau, M., Della-Libera, G., Dixon, B., Garg, P., Gravengaard, E., Gudgin, M., Hada, S., Hallam-Baker, P., Hondo, M., Kaler, C., Lockhart, H., Martherus, R., Maruyama, H., Mishra, P., Nadalin, A., Nagaratnam, N., Nash, A., Philpott, R., Platt, D., Prafullchandra, H., Sahu, M., Shewchuk, J., Simon, D., Srinivas, D., Waingold, E., Waite, D., & Zolfonoon, R. (2004).

Web Services Secure Conversation Language (WS-SecureConversation), from ftp://www6.software.ibm.com/software/developer/library/ws-secureconversation.pdf

Anderson, S., Bohren, J., Boubez, T., Chanliau, M., Della-Libera, G., Dixon, B., Garg, P., Gravengaard, E., Gudgin, M., Hallam-Baker, P., Hondo, M., Kaler, C., Lockhart, H., Martherus, R., Maruyama, H., Mishra, P., Nadalin, A., Nagaratnam, N., Nash, A., Philpott, R., Platt, D., Prafullchandra, H., Sahu, M., Shewchuk, J., Simon, D., Srinivas, D., Waingold, E., Waite, D., & Zolfonoon, R. (2004). *Web Services Trust Language (WS-Trust)*, from ftp://www6.software.ibm.com/software/developer/library/ws-trust.pdf

Atkinson, B., Bellwood, T., Cahuzac, M., Clément, L., Colgrave, J., Corda, U., Czimbor, A., Dovey, M. J., Feygin, D., Garg, S., Gupta, R., Hately, A., Henry, B., Kawai, A., Macias, P., Manes, A. T., Riegen, C. v., Rogers, T., Srivastava, A., Thorpe, P., Triglia, A., Voskob, M., & Zagelow, G. (2003). *UDDI Version 3.0.1 - UDDI Spec Technical Committee*

Specification 14 October 2003, from http://uddi.org/pubs/uddi-v3.0.1-20031014.htm

Bajaj, S., Box, D., Chappell, D., Curbera, F., Daniels, G., Hallam-Baker, P., Hondo, M., Kaler, C., Langworthy, D., Malhotra, A., Nadalin, A., Nagaratnam, N., Nottingham, M., Prafullchandra, H., Riegen, C. v., Schlimmer, J., Sharp, C., & Shewchuk, J. (2004). *Web Services Policy Framework (WS-Policy)*, from ftp://www6.software.ibm.com/software/developer/library/ws-policy.pdf

Bajaj, S., Della-Libera, G., Dixon, B., Dusche, M., Hondo, M., Hur, M., Lockhart, H., Maruyama, H., Nagaratnam, N., Nash, A., Prafullchandra, H., & Shewchuk, J. (2003). *Web Services Federation Language (WS-Federation)*, 2003, from http://www-106.ibm.com/developerworks/webservices/library/ws-fed/

Ballinger, K., Box, D., Curbera, F., Davanum, S., Ferguson, D., Graham, S., Liu, C. K., Leymann, F., Lovering, B., Nadalin, A., Nottingham, M., Orchard, D., Riegen, C. v., Sedukhin, I., Shewchuk, J., Smith, B., Weerawarana, S., & Yendluri, P. (2004). *Web Services Metadata Exchange (WS-*

MetadataExchange), from http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-metadataexchange.pdf

Bilorusets, R., Bosworth, A., Box, D., Cabrera, L. F., Collison, D., Ferguson, D., Ferris, C., Freund, T., Hondo, M. A., Ibbotson, J., Kaler, C., Langworthy, D., Lewis, A., Limprecht, R., Lucco, S., Mihic, M., Mullen, D., Nadalin, A., Nottingham, M., Orchard, D., Samdarshi, S., Shewchuk, J., & Storey, T. (2004). *Web Services Reliable Messaging Protocol (WS-ReliableMessaging)*, from http://msdn.microsoft.com/webservices/understanding/specs/default.aspx?pull=/library/en-us/dnglobspec/html/ws-reliablemessaging.asp

Cabrera, L. F., Copeland, G., Feingold, M., Freund, T., Johnson, J., Kaler, C., Klein, J., Langworthy, D., Nadalin, A., Orchard, D., Robinson, I., Storey, T., & Thatte, S. (2004). *Web Services Atomic Transaction (WS-AtomicTransaction)*. Retrieved December 2004, 2004, from http://msdn.microsoft.com/library/en-us/dnglobspec/html/WS-AtomicTransaction.pdf

Chang, S., Chen, Q., & Hsu, M. (2003, November 03 - 06, 2003). *Managing Security Policy in Large Distributed Web Services Environment.* Paper presented at the 27th Annual International Computer Software and Applications Conference (COMPSAC'03), Dallas, Texas.

Cranor, L., Langheinrich, M., Marchiori, M., Presler-Marshall, M., & Reagle, J. (2002). *The Platform for Privacy Preferences 1.0 (P3P1.0) Specification*, from http://www.w3.org/TR/P3P/

Della-Libera, G., Dixon, B., Dusche, M., Furguson, D., Garg, P., Hahn, T., Hinton, H., Hondo, M., Kaler, C., Leymann, F., Lovering, B., Maruyama, H., Nadalin, A., Nagaratnam, N., Raghavan, V., & Shewchuk, J. (2003, July 2003). *Federation of Identities in a Web Services World*, from http://www-106.ibm.com/developerworks/library/ws-fedworld/

Dingledine, R., Freedman, M. J., & Molnar, D. (2000). *The Free Haven Project: Distributed Anonymous Storage Service*, from http://www.freehaven.net/paper/node6.html

Eastlake, D. (1999). *Domain Name System Security Extensions (RFC 2535)*, from http://www.rfc-archive.org/getrfc.php?rfc=2535

Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., & Ylonen, T. (1999). *SPKI Certificate Theory*, from http://www.ietf.org/rfc/rfc2692.txt

Endrei, M., Ang, J., Arsanjani, A., Chua, S., Comte, P., Krogdahl, P., Luo, M., & Newling, T. (2004a). *Patterns: Service-Oriented Architecture and Web Services* (1st ed.).

Endrei, M., Ang, J., Arsanjani, A., Chua, S., Comte, P., Krogdahl, P., Luo, M., & Newling, T. (2004b). *Patterns: Services Oriented Architectures and Web Services*.

Erl, T. (2004). *Service Oriented Architecture. A Field Guide to Integrating XML and Web Services.* (1st ed.). New Jersey: Prentice Hall PTR.

Evans, C., Chappell, D., Bunting, D., Tharakan, G., Shimamura, H., Durand, J., Mischkinsky, J., Nihei, K., Iwasa, K., Chapman, M., Shimamura, M., Kassem, N., Yamamoto, N., Kunisetty, S., Hashimoto, T., Fujitsu, T. R., & Nomura, Y. (2003). *WS-Reliability*, from otn.oracle.com/tech/webservices/htdocs/spec/WS-ReliabilityV1.0.pdf

Farrell, S., Reid, I., Lockhart, H., Orchard, D., Sankar, K., Adams, C., Moses, T., Edwards, N., Pato, J., Blakley, B., Erdos, M., Cantor, S., Morgan, R. B., Chanliau, M., McLaren,

C., Knouse, C., Godik, S., Platt, D., Moreh, J., Hodges, J., & Hallam-Baker, P. (2003, 2 September 2003). *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1*, from http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf

Geuer-Pollmann, C. (2002, 2002). *XML Pool Encryption.* Paper presented at the Workshop on XML Security, Fairfax, VA.

Grandison, T., Sloman, M., & College, I. (2000). *A Survey of Trust in Internet Applications* (Survey): IEEE.

Housley, R., Ford, W., Polk, W., & Solo, D. (1999). *Internet X.509 Public Key Infrastructure Certificate and CRL Profile (RFC 2459)*, from http://www.ietf.org/rfc/rfc2459.txt

IBM, & Microsoft. (2002a, April,7 2002). *IBM and Microsoft. Security in a Web Services World: A Proposed Architecture and Roadmap - technical whitepaper 7 April 2002*, from http://msdn.microsoft.com/ws-security/

IBM, & Microsoft. (2002b). *Security in a Web Services World: A Proposed Architecture and Roadmap*, from http://www-106.ibm.com/developerworks/webservices/library/ws-secmap/

IBM, & Microsoft. (2003). *Federation of Identities in a Web Services World*, 2004, from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-federation-strategy.asp

IBM, Microsoft, & TIBCO. (2004). *Web Services Reliable Messaging Protocol (WS-ReliableMessaging)*, from ftp://www6.software.ibm.com/software/developer/library/ws-reliablemessaging200403.pdf

Imamura, T., Dillaway, B., & Simon, E. (2002, 10 December 2002). *W3C XML Encryption Syntax and Processing - W3C Recommendation 10 December 2002*, from http://www.w3.org/TR/xmlenc-core/

Internet2/MACE. (2002, May 2, 2002). *Shibboleth-Architecture DRAFT v05*, 2003, from http://shibboleth.internet2.edu/draft-internet2-shibboleth-arch-v05.html

Lampson, B., Abadi, M., Burrows, M., & Wobber, E. (1992). Authentication in Distributed Systems: Theory and Practice. *ACM Transactions on Computer Systems, 10*(4), 265-310.

LibertyAllianceProject. (2003, March, 2003). *Liberty Alliance Project. Introduction to the Liberty Alliance Identity*

Architecture, from http://www.projectliberty.org/resources/whitepapers/LAP%20Identity%20Architecture%20Whitepaper%20Final.pdf

Lockhart, H., Moses, T., Prodromou, E., Erdos, M., Morgan, R. B., McLaren, C., Mishra, P., & Hodges, J. (2003). *Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V1.1 - OASIS Standard, 2 September 2003*, from http://www.oasis-open.org/committees/download.php/3404/oasis-sstc-saml-sec-consider-1.1.pdf

NIST. (2003, 4 April 2003). *National Institute of Standards and Technology. Role-based Access Control - Draft 4 April 2003*, from http://csrc.nist.gov/rbac/rbac-std-ncits.pdf

OASIS. (2004, 6 April 2004). *Web Services Security (WS-Security) - Specification 6 April 2004*, from http://www-106.ibm.com/developerworks/webservices/library/ws-secure

OASISWebServicesSecurityTC. (2004, 6 April 2004). *Web Services Security (WS-Security) - Specification 6 April 2004*, from http://www-106.ibm.com/developerworks/webservices/library/ws-secure

Pfitzmann, A., & Köhntopp, M.*Anonymity, Unobservability, and Pseudonymity*, from http://www.freehaven.net/anonbib/cache/terminology.pdf

Polivy, D. J., & Tamassia, R. (2002, Nov. 22, 2002). *Authenticating Distributed Data Using Web Services and XML Digital Signatures.* Paper presented at the ACM Workshop on XML Security.

Saltzer, J. H., Reed, D. P., & Clark, D. D. (1984). End-to-End Argument in System Design. *ACM Transactions on Computer Systems, 2*(4), 277-288.

TC, X. (2004, 13 February 2004). *XACML Profile for Role Based Access Control (RBAC)*, 2004, from http://docs.oasis-open.org/xacml/cd-xacml-rbac-profile-01.pdf

Todd, S., Parr, F., & Conner, M. H. (2002, 1 July 2001). *A Primer for HTTPR. An overview of the reliable HTTP protocol.*, from http://www-106.ibm.com/developerworks/webservices/library/ws-phtt/

W3C. (2004, 11 February 2004). *Web Services Architecture*, from http://www.w3.org/TR/ws-arch/

W3CWebServicesArchitectureWorkingGroup. (2004, 11 February 2004). *Web Services Architecture*, from http://www.w3.org/TR/ws-arch/

W3CXMLProtocolWorkingGroup. (2003, 24 June 2003). *W3C SOAP Version 1.2 Part 0: Primer*, from http://www.w3.org/TR/2003/REC-soap12-part0-20030624/

WS-I. (2004). *Basic Profile Version 1.0*, from http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html

Zhang, L.-J., Zhang, J., & Chung, J.-Y. (2004, September 13-15). *An Approach to Help Select Trustworthy Web Services.* Paper presented at the E-Commerce Technology for Dynamic Business, IEEE International Conference on CEC-East'04, Beijing, China.

Zimmerman, P. (1994). *PGP User's Guide.* Cambridge: MIT Press.