Seguridad para la Autentificación, Cifrado y Firma en la Ejecución de Servicios Web Compuestos

■ Marcos Grillo
email: 13-89388@usb.ve
Universidad Simón Bolívar
Caracas, Venezuela

■ Wilmer Pereira
email: wpereira@ucab.edu.ve
wpereira@usb.ve
Universidad Católica Andrés Bello
Universidad Simón Bolívar
Caracas, Venezuela

■ Yudith Cardinale
email: yudith@ldc.usb.ve
Universidad Simón Bolívar
Caracas, Venezuela

Fecha de Recepción: 26 de Julio 2015
Fecha de Aceptación: 10 de Septiembre de 2015

RESUMEN

En un sistema distribuido se requieren múltiples mecanismos de seguridad como: verificación de la identidad de los servidores (autentificación), detección de alteraciones de los resultados (integridad) o aseguramiento de la privacidad de la información (confidencialidad). Para el caso de la composición de servicios web, en donde varios servicios web se articulan para cumplir el requerimiento de un usuario, un intruso puede alterar la información intercambiada entre los servidores. Esto se puede evitar, por una parte, cifrando las respuestas, mediante negociado de una clave de sesión, para que el

cliente no reciba, por ejemplo, información fraudulenta o publicidad difamatoria. Por otra parte, valiendose de la infraestructura de claves públicas y privadas, es posible asegurar la autenticidad de los servidores Web mediante certificados y validar la autoría del mensaje para que se asuman los compromisos de la información transmitida usando firmas digitales. Asî en este trabajo se propone el uso de la infraestructura de certificación digital para ofrecer seguridad en el intercambio de información en sistemas distribuidos que realicen composición de servicios web, permitiendo: integridad, confidencialidad, autentificación mutua y evitar el rechazo de autoría por la información transmitida (no repudio).

Palabras Claves–Composición de Servicios Web, Certificados Digitales, Autentificación, Cifrado, Firma Digital.

I. INTRODUCIÓN

Los Servicios Web (SW) han sido la base para la integración de la arquitectura orientada a servicio como plataforma de desarrollo para aplicaciones de distintas organizaciones. El objetivo es tomar ventaja de la existencia de servicios web bien conocidos para satisfacer requerimientos. La Composición de Servicios Web (CSW) surge como una facilidad para el cliente, permitiendo la integración de diversos SW y así mediante esta infraestructura, dar respuesta a problemas más complejos que no pudieran ser satisfechos con la consulta de un solo servicio Web [1]. La pretención es saber cómo y cuáles respuestas de los Servicios Web deben ser combinadas para obtener el resultado deseado.

En la Composición de Servicios Web se tienen dos fases, una primera de definición para determinar el conjunto de SW que se orquestarán para satisfacer el requerimiento de un cliente; seguida de una fase de ejecución durante la cual se realizan las invocaciones de dichos SW en el orden establecido en la fase de definición [2]. La fase de definición es llevada a cado por un compositor y la de invocación a los SW es llevada a cabo por un motor de ejecución.

La ejecución de los SW efectivamente puede hacerse secuencialmente o en paralelo dependiendo del flujo de dependencia entre los distintos SW. En algunos casos, las salidas (resultados de la invocación) de un SW son la entrada (datos de invocación) de otro SW, por lo que esto define una ejecución secuencial. Si no hay relación de dependencia, la ejecución de los distintos SW puede ser en paralelo y consolidadas por el motor de ejecución.

Por otro lado, existen dos esquemas ampliamente utilizados para implementar los motores de ejecución. El primero es el esquema centralizado, donde se tiene un coordinador encargado de la invocación de todos los SW para la composición. El segundo es el esquema distribuido, que consiste en distintos nodos del motor de ejecución encargados de ejecutar una parte de la composición. Así cada motor de ejecución debe transmitir las salidas de sus SW a otros motores de ejecución para que invoquen sus respectivos SW.

Proponer un motor de ejecución como un sistema distribuido tiene varios problemas que no se presentan con un motor centralizado. Uno de ellos es que los distintos nodos del motor de ejecución se transmiten información que debe ser compuesta para dar una respuesta unificada. Al estar distribuido por la red, no se puede descartar que algún ente intervenga de alguna forma la ejecución para alterar el resultado final de la composición. Por ejemplo, un tercero mal intencionado podría hacer un ataque de suplantación de identidad para reemplazar la respuesta correcta o un ataque de hombre en

el medio para interceptar el mensaje e igualmente alterar su contenido. Hoy en día hay diversos mecanismos de seguridad basados en la certificación digital que se pueden aplicar para prevenir estos ataques. La infraestructura que se vale de los certificados digitales, conocida como PKI (Private Key Infrastructure), ofrece varios servicios de seguridad convenientes para la Composición de Servicios Web.

En este trabajo nos centramos en tres mecanismos de seguridad para proteger la información que se intercambian los motores de ejecución. En primer lugar agregar la posibilidad de ocultar la información en tránsito ante terceros no autorizados mediante cifrado. En segundo lugar realizar autentificación mutua, mediante certificados digitales, para que se verifique la identidad de los nodos del motor de ejecución que se comunican. Por último introducir un medio de verificación de la autenticidad de la información mediante una firma digital de las respuestas a componer para así asegurar la integridad y evitar el repudio de las transacciones.

El artículo está organizado, inicialmente, con dos secciones que presentan el marco de teórico que sustenta el trabajo: la composición de servicios Web y los principios de seguridad bajo la infraestructura de certificación digital. Justo después se describe la arquitectura de servicios web compuestos seguros inspirado en el uso de certificados digitales y finalmente se presenta un caso de estudio para mostrar las ventajas que ofrece al usuario una arquitectura de composición de servicios Web segura.

II. SERVICIOS WEB COMPUESTOS

La Composición de Servicio Web (CSW) es una combinación de varios SW para producir servicios variados y específicos que satisfacen requerimientos complejos de un usuarios [2]. Hay dos procesos esenciales en los CSW: la composición y la ejecución.

Por un lado, la composición consiste en definir cómo y cuáles SW se combinan para obtener los resultados deseados por el usuario. Un CSW puede ser representado con estructuras como workflows, grafos o redes de petri, para indicar, por ejemplo el flujo de control, el flujo de datos, el orden de ejecución de los SW y el comportamiento de los SW. La estructura que representa un CSW puede ser generada manual o automáticamente. Los usuarios pueden especificar manualmente cómo las funcionalidades de los SW pueden combinarse o un agente compositor puede decidir automáticamente cuales y como se combinan los SW, de acuerdo al requerimiento del usuario. En ambos casos, la ejecución del CSW está a cargo de un motor de ejecución.

Los SW se describen de acuerdo a sus funcionalidades (e.g., parámetros de entrada y salida, precondiciones, efectos) y condiciones de QoS (Quality of Services), cuyos valores describen la eficacia y eficiencia de la ejecución de los SW disponibles (e.g., tiempos de respuesta, costos, confiabilidad, confianza, capacidades de conexión).

Los parámetros funcionales y de QoS son importantes al momento de decidir cuales SW seleccionar para obtener una eficiente y efectiva composición [3].

Por otro lado, la ejecución de un servicio Web compuesto implica la invocación de todos los SW que lo componen de acuerdo al flujo de ejecución establecido. Si la ejecución es secuencial, algunos SW no

pueden ser invocados hasta que previos SW hayan finalizado, ya sea porque requieren los resultados de los SW previos o por restricciones de control secuencial impuesta. Cuando la ejecución es paralela, varios SW pueden ser invocados simultáneamente dado que no hay dependencias de datos o de flujo de control entre las respuestas de los SW.

El motor de ejecución encargado de dichas invocaciones puede ser implementado siguendo un enfoque centralizado, donde se tiene un solo coordinador central que invoca los SW que conforman el SWC, o distribuido, donde la ejecución se realiza con la colaboración de diversos nodos del motor de ejecución responsables de la invocación de uno o más SW y de transferir sus resultados de la ejecución a otros nodos del motor de ejecución encargados de la invocación de otros SW que componen la respuesta [4].

III. MECANISMOS DE SEGURIDAD

La criptografía moderna, tal como la conocemos actualmente, sentó sólidas bases a comienzos del siglo pasado inicialmente como producto de la confidencialidad que exigen los conflictos bélicos. En aquel momento y aún actualmente, se usan los algoritmos basados en una clave, también conocidos como los algoritmos de clave simétrica. Estos se valen de una contraseña compartida por ambos interlocutores, con la cual, el emisor cifra y el receptor retoma la misma contraseña para descifrar.

Formalmente los algoritmos de clave simétrica se sustentan en manipulaciones sobre el texto en claro, usando técnicas de substitución y transposición en secuencias de caracteres. A pesar de su probada fortaleza, por si solos tiene el problema de la distribución segura, a través de la red, de esa clave compartida.

Motivados por esta y otras limitaciones y gracias al esfuerzo de académicos y especialistas en criptografía, surgen los algoritmos basados en dos claves: una pública a difundir entre los interlocutores y una privada que el usuario guarda fuertemente protegida.

El cifrado con estos algoritmos se logra cuando el emisor utiliza la clave pública del receptor y el destinatario descifra con su privada. Ahora basta con tener la pública del destinatario, que puede enviarse por la red o publicarse en claro en un servidor, para poder enviar información confidencial de manera segura. Sin embargo, desafortunadamente los algoritmos de clave pública son cerca de 1000 veces más lentos que los algoritmos de clave simétrica. Por lo tanto al cifrar gran cantidad de informaciôn con estos algoritmos, los tiempos de cifrado y descifrado se hacen inaceptables para aplicaciones en tiempo real. En consecuencia, deben usarse los algoritmos de clave simétrica por ser más rápidos ... no obstante persiste el problema de negociado de una clave simétrica. Afortunadamente este problema se resuelve facilmente negociando la clave simétrica por intermedio de los algoritmos de clave pública. La clave simétrica es propuesta aleatoriamente por uno de los interlocutores. Así con esa clave conocida de ambos interlocutores, se cifra la informaciôn a proteger con los algoritmos más eficientes.

Por otro lado, aunque la clave pública debe estar en claro, para evitar ataques de suplantación de identidad, debe estar avalada por un tercero de confianza. Con ello, toda clave pública avalada sirve ahora como un medio para autentificar al propietario. Para lograr esta verificación de la identidad de un usuario, basta con que el emisor entregue al destinatario su clave pública. Si la clave está avalada por un tercero confiable queda verificada la identidad del emisor. De este modo, nace la idea de las Autoridades de Certificación (AC) como los entes confiable

que verifican la identidad del dueño de una clave pública, dando fe de su identidad. Una AC avala la clave pública de sus clientes, utilizando una propiedad de la clave privada que por ser confidencial y en principio única, se puede construir un mecanismo para autentificar información. Es decir, al cifrar un texto con la privada, por ejemplo la clave privada de la AC, se da fé de la información cifrada. En este caso, es suficiente con que el receptor tenga la clave pública de la AC para estar seguro que la información fue vista y aprobada por la AC. Si además se utilizan funciones de hash para reducir su tamaño, se define un mecanismo rápido y eficiente para firmar documentos digitales.

En principio, la firma se establece calculando hash o el resumen o compendio de la información, utilizando algoritmos como SHA-256 o MD5, y luego se cifra utilizando la clave privada del emisor. El resultado se envía junto con la información original. Por su parte el receptor teniendo el certificado con la clave pública del emisor, calcula el resumen de la información recibida, se extrae el compendio original de la información enviada con la clave pública y finalmente se comparan los resumenes. Si son iguales, entonces no hubo alteraciones y se tiene certeza que la respuesta fue generada por el emisor.

En caso contrario, si no coinciden los resumenes, se rechaza la información enviada y eventualmente se solicita retransmisión.

Esto además de garantizar la integridad, resguarda la autenticidad de la información por parte del emisor e impide que que este repudie o niegue la autoría de la información enviada.

Ahora se dispone de un mecanismo para avalar información que es utilizado por las AC. Ellas se valen de la firma para certificar la identidad de los usuarios que acuden a ellos para que den aval de su identidad. Para las AC, los documentos que portan la clave pública de sus clientes avalados con su firma son los conocidos como certificados digitales.

Finalmente todas estas estrategias: cifrado con algoritmos de clave simétrica, autentificación de usuarios por medio de certificados digitales y firma para avalar la autoría de información enviada, permiten lograr una composición de servicios Web segura.

Todos estos mecanismos son bien conocidos por lo que, adicionalmente, nuestra propuesta ofrece la posibilidad de manejar certificados por cada transacción entre los SW evitando los inconvenientes de la revocación. Para ello nos valemos de una Autoridad de Certificación propia a nuestra infraestructura que denominaremos Autoridad de Certificación Interna (ACI) autofirmada. Con ella se crean constantemente certificados que vencen inmediatamente después que el SW envía su respuesta, como se hace en algunas plataforma de Grids Computacionales. También los clientes a esta plataforma deben tener certificados que se generan en tiempo real al momento de hacer el registro. Esta funcionalidad evita el retraso que implica la creación off-line de certificados para los usuarios como Autoridades de Certificación Externas como Verisign, Start SSL, Comodo, GlobalSign, etc. Además no se necesitan Autoridades de Registro ya que la infraestrucutura debe ser lo más abierta posible sin descuidar aspectos básicos de seguridad.

IV. TRABAJOS RELACIONADOS

En principio, la Composición de Servicios Web seguros podría ser cuestionable ya que, en principio, la información que ofrecen los SW es gratuita y de libre acceso a cualquiera que la solicite. Sin embargo cuando se plantea el escenario de la composición de servicios no centralizada surgen los problemas de alteración de las respuestas de los servidores para atentar contra el buen funcionamiento de la infraestructura. En otras palabras, sin mecanismos de seguridad, un intruso podrîa realizar, por ejemplo, un ataque de suplantaciôn de identidad de un WS, y enviar información fraudulenta.

Uno de los trabajos más conocidos en seguridad para WS lo realizó OASIS (Organization for the Advancement of Structured Information Standards) que es un consorcio internacional, sin fines de lucro, que orienta el desarrollo de los estándares de comercio electrónico y servicios web.

Especificamente OASIS definió WS-Security (WSS) [5] el cual es un protocolo de comunicaciones para montar seguridad en los WS. La versión 1.1. provee soporte para múltiples formatos de mensajes seguros, con autentificación, control de acceso, confidencialidad y firma usando distintas estrategias:

-Certificados digitales bajo TLS siguiendo el estandar PKI [6] o

-Cifrado y autentificación con algoritmos de clave simétrica bajo la plataforma Kerberos o

-SAML (Security Assertion Markup Language) que permite autentificación y autorización dentro de XML.

-Definición de nombre de usuario y clave

WSS ofrece gran variedad de algoritmos de cifrado con diferentes funcionalidades. Esto le da gran versatilidad aunque, por ahora, implica la instalación del protocolo tanto en los WSS como en los navegadores.

WSS también permite, por intermedio de SAML, un mecanismo de autorización, es decir, control de acceso para los clientes que desean acceder a los diferentes recursos. En cambio nuestra propuesta funciona con el protocolo que tienen por defecto todos los navegadores y servidores Web como SSL y TLS. Además el prototipo desarrollado en este trabajo funciona con certificados digitales temporales que son generados dinámicamente. En consecuencia, simplifica los procesos de autentificación pues hace innecesario el uso de autentificación mediante login y password. Más aún no se requiere de una AC externa a la cual se debe pagar anualmente por la creación de certificados lo que hace nuestra propuesta más económica y flexible.

El enfoque que definimos no tiene previsto el control de acceso, ya que se supone que los WS son de libre acceso y cualquier puede acceder a la información de cada WS a condición de autentificarse. Tampoco se puede acoplar con kerberos ni con SAML.

Por otro lado, un intento de ofrecer seguridad en servicios Web la propone Qi Zhang en [7] con la utilización de certificados digitales basados en el algoritmo RSA para, por ejemplo, autentificar la identidad de una aplicación en red o servicio Web. Utilizando este mecanismo con certificados, es posible cifrar la comunicación entre los elementos de la arquitectura y establecer una cadena de certificación que autentifique a cada uno de los motores de ejecución.

Nuestro trabajo, a diferencia del anterior, ofrece más funcionalidades (cifrado y

firma) y, con la creación de certificados mediante la Autoridad de Certificaciôn Interna, se evita que el usuario espere por un certificado avalado por Autoridades de Certificación Externas.

Otra propuesta la constituye el trabajo de [8] que pretende definir restricciones de seguridad desde el punto de vista de la Web semántica con instrucciones usando el lenguaje ontológico OWL. Allí se definen básicamente condiciones de autentificación y confidencialidad para proteger un protocolo definido.

En nuestro caso, el protocolo que sustenta nuestra propuesta es SSL que tiene una probada eficacia en los procesos de autentificación y cifrado de información.

V. ARQUITECTURA

La propuesta se centra en construir una arquitectura segura para un motor de ejecución distribuido, conformado por varios< nodos, que se ejecutan en computadores conectados a través de una WAN. La composición es el insumo previo para el motor construido como una red de Petri.
Especificamente describiremos el mecanismo de cifrado para asegurar confidencialidad, autentificación mutua y la firma de las respuestas de los motores de ejecución para el resguardo de la integridad en las información suministrada de los SW.

V-A. Cifrado en canales de comunicación

Esta arquitectura se sustenta en el protocolo SSL que ofrece algunas funcionalidades que facilitan el desarrollo de nuestro prototipo de motor de ejecución seguro. A su vez usamos el protocolo HTTPS1 para el cual es imprescindible el uso de certificados digitales.
Inicialmente se define una cadena de confianza partiendo de un certificado autofirmado que juega el rol de autoridad de certificación interna y con cuya clave privada, se firma cada certificado de los componentes del sistema, es decir cada nodo del motor de ejecución. Los certificados de cada nodo se crean dinámicamente, por cada transacción, y vencen al momento que el WS envía su respuesta.
Por defecto el protocolo SSL negocia una clave simétrica valiéndose de la clave pública de uno de los nodos del motor de ejecución. Esto se realiza durante el handshake donde se intercambian los certificados digitales conteniendo las claves públicas de ambos nodos. Un nodo propone una clave simétrica generada aleatoriamente que se cifra con la clave pública del otro nodo, negociando así una clave simétrica para cifrar los datos a intercambiar.
El cifrado es obligatorio entre los nodos del motor de ejecución y opcional para las respuestas que llegan al cliente. En caso de que el cliente no solicite cifrado, el tiempo total de ejecución se reduce ya que el cliente estimó que su peticiôn no requería confidencialidad.

V-B. Autentificación mutua

Para la autentificación, se propone un mecanismo más fuerte que el definido por defecto en SSL que sólo exige autentificación obligatoria del servidor mediante entrega de certificado digital al cliente.
En nuestra propuesta de CSW, los diferentes nodos del motor de ejecución, por tener roles similares, del mismo estilo de las arquitecturas peer to peer, utilizan un esquema de doble autentificación, es decir, que ambos nodos están obligados a instalar e intercambiar sus certificados.

1HTTP Over TLS http://tools.ietf.org/html/rfc2818 tomado en Abril de 2015 protocolo de comunicación cifrado

Tambien se entregan certificados a los clientes que hacen uso de la infraestructura de servicios Web compuestos. Así se verifica la identidad de todos los entes involucrados y se evita la suplantación de identidad y por ende falsificación de las respuestas. Esto es posible gracias a la cadena de seguridad descrita en la Figura 1.

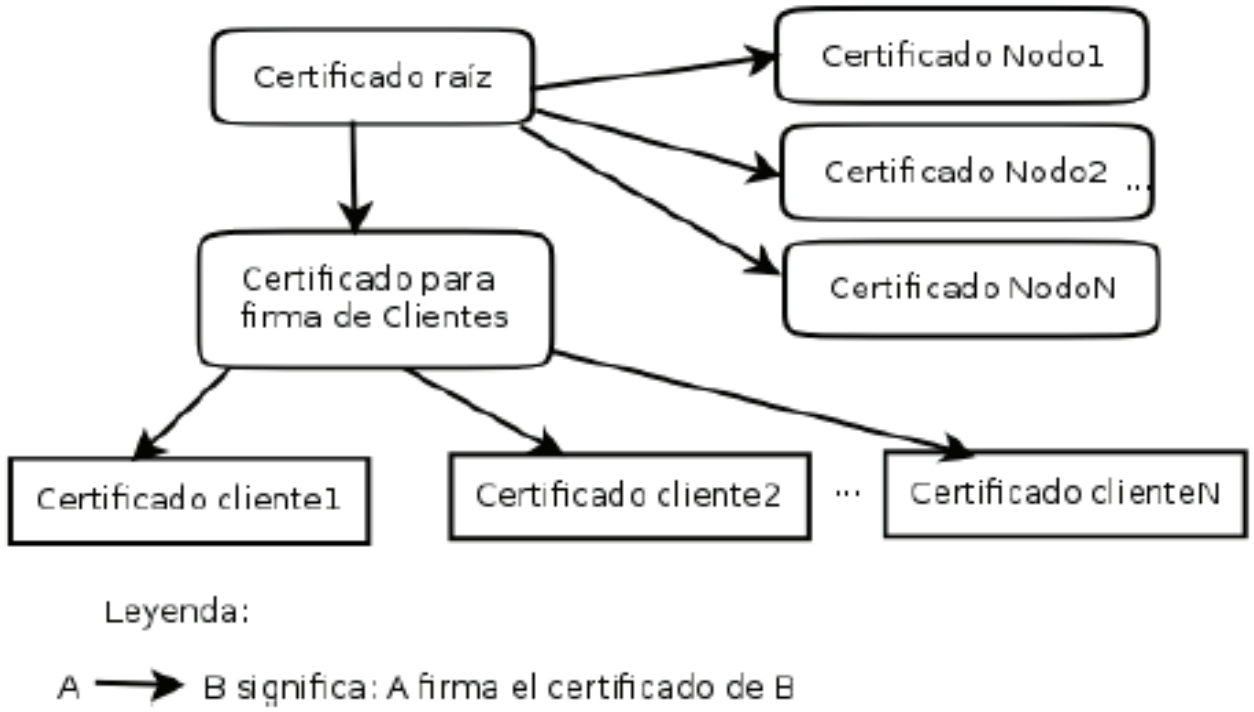

Figura 1: Cadena de seguridad para firma de certificados

La infraestructura tiene el certificado raíz autofirmado o AC interna en el compositor. Este es quien firma los certificados por transacción para los nodos del motor de ejecución y también los certificados con fecha de vencimiento para los clientes. Esa fecha es independiente de cuantas transacciones realice el cliente. Las normas clásicas definen certificados para personas naturales y jurídicas generalmente por un año pero esto es configurable para esta plataforma de composición de servicios Web segura.

En consecuencia, cuando los clientes se conectan por primera vez al compositor emite un formulario que el usuario debe completar. La AC interna es quien firma ese certificado de cliente y un script lo instala en el navegador del usuario. Una vez cumplida esta fase inicial en el cliente, se hace la composición de su requerimiento que es enviado al motor de ejecuciôn. Si el cliente se conecta de nuevo por otra solicitud, ya no necesita llenar de nuevo el formulario y puede continuar hasta el vencimiento del certificado. En algún momento, cuando el certificado no sea válido, recibirá advertencias que le sugieren que debe pasar de nuevo al proceso de registro.

Las advertencias no impiden el cifrado del canal pero indican que hay un fallo en el proceso de autentificación.

Así se autentifica automáticamente con el intercambio de certificados sin requerir del uso de una clave pues el compositor funciona bajo el mecanismo de autentificaciôn por certificados del protocolo HTTPS.
Las claves públicas de los nodos del motor de ejecución, están en los certificados temporales y tienen asociados una clave privada que guardan locamente. Esta clave es necesaria para que el nodo del motor de ejecución firmar las respuestas que haga llegar a otros nodos y así continuar con la composición del requerimiento.

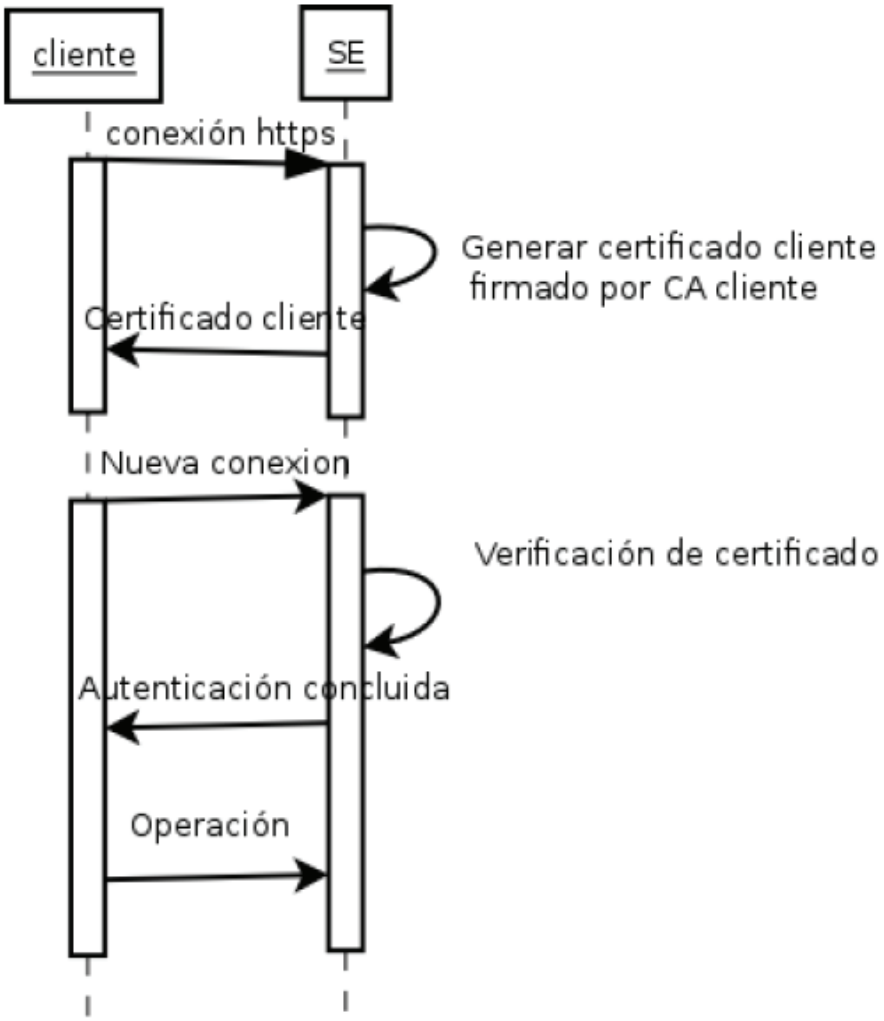

Figura 2: Diagrama de interacción para generación de certificados de clientes

Aunque no está previsto en el prototipo desarrollado, el cliente también puede firmar sus solicitudes para dar garantía al compositor de quien efectivamente hizo la solicitud.
En la Figura 2 se muestra un diagrama de secuencia para el proceso de generación de certificados tanto para los nodos del motor de ejecución como para los clientes.

V-C. Integridad en las respuestas

Por último para asegurar que la respuesta proviene de un nodo autorizado y evitar ataques de repetición, justo antes de enviar la información, el nodo del motor de ejecución agrega a la información procesada la fecha de culminación de la ejecución y firma la respuesta fechada con su clave privada.

Cada nodo verifica la firma del nodo que le entregó la respuesta parcialmente compuesta. Esta verificación es posible ya que cada nodo posee la clave pública del nodo con el que se comunica a través del certificado digital temporal. Justo después de la comprobación de identidad, firma su respuesta y la entrega fechada al siguiente nodo del motor de ejecución.

La firma además de autentificar al nodo que procesó la información a componer, es una verificación de integridad de los datos enviados.

V. CASO DE ESTUDIO

En esta sección se ilustrará con un ejemplo el funcionamiento de la estrategia de seguridad propuesta. Para probar dicha infraestructura, se implementó un prototipo funcional usando javascript con el framework nodejs. Los mecanismos de seguridad implantados son:

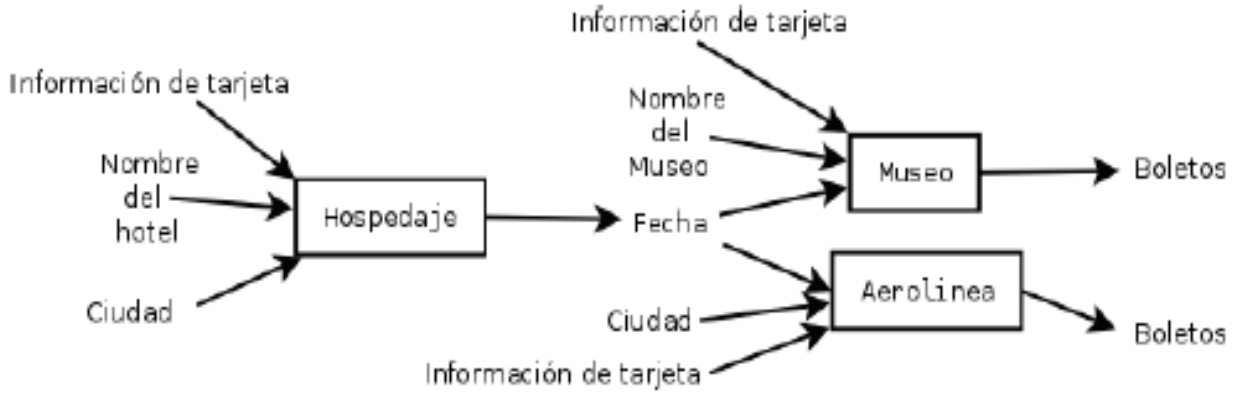

Figura 3: Composición de SW del caso de estudio

1. Cifrado con clave simétrica negociada a través de algoritmos de clave pública
2. Autentificación mutua mediante intercambio de certificados digitales firmados por una autoridad interna y emitidos temporalmente (nodos del motor de ejecución) y con fecha de vencimiento (clientes)
3. Firma digital de los mensajes con fecha enviados para autentificar los mensajes y evitar el rechazo de transacciones.

Supongamos que una persona desea comprar boletos de avión, reservar un hotel y visitar un museo emblemático de cierta ciudad y para ello suministra el nombre del hotel, el nombre del museo, el nombre de la ciudad y su información de tarjeta de crédito.
Asumiendo que en la ciudad hay servicios web que dan la disponibilidad de hoteles, transporte y ventas de boletos al museo, el compositor establece una petición de servicios web que satisfaga las restricciones del cliente.

Supongamos que los SW necesarios para cumplir la petición se encuentran en el Cuadro I:

Cuadro I: Servicios del caso de estudio

SW	Entrada	Salida
hospedaje	información de tarjeta, ciudad, nombre del hotel	fecha
museo	información de tarjeta, nombre del museo, fecha tentativa	boleto
aerolinea	información de tarjeta, ciudad, fecha tentativa	boleto

El servicio de hospedaje recibe el nombre del hotel, chequea la disponibilidad y finalmente emite una fecha posible de reserva. Los servicios de museo y aerolinea reciben una fecha tentativa de estadía del usuario, en paralelo revisan su disponibilidad y emiten una fecha que se apegue a los parámetros recibidos.

En la Figura 3 se muestra una representación de la composición de servicios que puede resolver la solicitud a partir de los datos del usuario. En [3] se propone un algoritmo para realizar la composición de forma automática. Los envíos entre los nodos del motor de ejecución son cifrados, con autentificación mutua entre ellos y con firma de las respuestas fechadas.

Los pasos realizados por el motor de ejecución, dado el requerimiento estructurado por el compositor, e independientemente de los aspectos de seguridad, se llevan a cabo de la siguiente manera:

1. Un nodo del motor de ejecución recibe la composición del CSW y hace la llamada al servicio Hospedaje.

2. El resultado del SW de Hospedaje, se envía a dos nodos del motor de ejecución que en paralelo buscaran el pasaje de la aerolínea y el boleto para ingresar al museo.

3. El motor de ejecución envía la respuesta al cliente.

Durante el proceso descrito, si el cliente decide pagar, envía la información de su tarjeta de crédito levantado una página o aplicación independiente del CSW. Así pagará la habitación, el pasaje y el boleto usando algún mecanismo seguro, probablemente montado sobre SSL, fuera de la infraestructura CSW.

Sin duda el número de la tarjeta de crédito es la información más importante del cliente que debe ser estrictamente protegida aunque no es parte del proceso de composición de servicios web seguro propuesto.

El hecho de que el cliente se registre y solicite un certificado con fecha de vencimiento, evita que maneje sesiones con login y password, que es más cercano al proceso tradicional de composición de servicios Web sin seguridad. Es decir, con certificados digital en ambos interlocutor simplifica el proceos de autentificación y la hace transparente al cliente.

Cuando el cliente realiza la solicitud del certificado, los pasos son:

1. Contacta al compositor y este verifica si el cliente ya tiene certificado para la plataforma. En caso contrario, solicita datos al cliente para su registro, firma un certificado con la AC interna y le devuelve al cliente un script que instalará el certificado en su navegador.

2. Busca un nodo del motor de ejecución y levanta una conexión sobre SSL con autentificación mutua.

3. Negocia una clave simétrica para cifrar la información con la respuesta del nodo del motor de ejecución. Esto último es opcional para el cliente. En caso de cifrar se sigue el procedimiento estandar de SSL.

4. Recibe la respuesta cifrada con la clave simétrica negociada y firmada con la clave privada del nodo. Descifra el contenido del mensaje, constata la veracidad de la firma gracias al certificado del nodo del motor de ejecución.

El nodo del motor de ejecución, por su parte, debe tener 3 puertos habilitados. El primero para certificación donde se generan los certificados temporales para los nodos del motor de ejecución y certificados para los clientes. El segundo que verifique la doble autentificación. El tercer puerto donde se reciben solicitudes de ejecución de CSW cifradas y firmadas.

Después que el cliente envía la solicitud y ya tiene un certificado, el proceso sigue así:

1. Se recibe una petición por el puerto de certificación que genera un certificado temporal para los nodos sólo para ese requerimiento (por transacción) firmado por la AC interna. A continuación envía tanto el certificado solicitado como el certificado de la AC interna para verificar las identidades de los nodos del motor de ejecución y verificar las firmas.

2. Se recibe una petición de ejecución por el puerto autentificación y se verifica que el certificado del cliente o de los nodos estén firmados por la AC interna.

3. El nodo del motor de ejecución consulta al SW, cifra, coloca fecha y firma la respuesta que envía al otro nodo o directamente al compositor.

Las peticiones de los clientes siempre se reciben firmadas aunque pueden estar cifradas o no. En cambio, la información que llega a los nodos del motor de ejecución siempre está cifrada y firmada. Por último la verificación de las firmas se logra gracias a los certificados que contienen las claves públicas que son necesarias para constatar que la firma es realizada por quien envió la información.

VII. CONCLUSIÓN Y TRABAJO FUTURO

Un motor de ejecución de CSW modelado como un sistema distribuido por lo general presenta un problema que los motores centralizados no tienen y es la vulnerabilidad antes diversos tipos de ataques (suplantación de identidad y hombre en el medio). Con los mecanismos propuestos en este trabajo se detecta la alteración de la información circulante y se evita la captura de la información confidencial gracias a la firma y el cifrado respectivamente. Con ellos se detecta la alteración de los mensajes y se asegurar la privacidad de los datos. Para lograr lo anterior es imprescindible el uso de certificados digitales agregando además un esquema de doble certificación o autentificación mutua.

Evitamos además los ataques de repetición ya que se fechan las solicitudes.

Evidentemente para que esto sea posible, es necesario que los distintos computadores tengan sus relojes sincronizados.

Por otro lado, podemos adaptar WSS a nuestra propuesta y esto nos ofrece todas las ventajas de distintos enfoque de cifrado, autentificación, integridad y no repudio. Por ejemplo, podriamos incluir control de acceso a partir del protocolo SAML. No obstante, nuestro enfoque utiliza protocolos bien conocidos para conectarse con los SW vía SSL. Aunque esto lo hace menos flexible, tenemos una clara ventaja de portabilidad en esta propuesta. Finalmente como WSS está concebido para ser adaptado a distintas aplicaciones, también podemos adoptarlo para ser usado por nuestra infraestructura y así, por ejemplo, acoplar Kerberos a nuestro modelo.

Por último, es claro que hay ataques para los cuales esta infraestructura no protege como por ejemplo la alteración sistemática de la información. Es decir, si el atacante intercepta todo el tráfico y cambia permanentemente la información cifrada y firmada, el emisor pide retransmisión pues detecta el ataque. Si al reenviarse la información el atacante modifica sin cesar todo reenvío, el receptor se verá impedido de tener la información solicitada. Esta es una suerte de ataque de negación de servicio. Actualmente estamos evaluando distintas estrategias para evitar estos ataques. También estamos estudiando otros ataques y las medidas a aplicar ante: peticiones no solicitadas, troyanos en los SW, etc.

VIII. REFERENCIAS

[1] R. Angarita, Y. Cardinale, and M. Rukoz, “Reliable composite web services execution: Towards a dynamic recovery decision,” Electronic Notes in Theoretical Computer Science, vol. 302, pp. 5–28, 2014.
[2] Y. Cardinale, J. El Haddad, M. Manouvrier, and M. Rukoz, “Cpn-tws: a coloured petri-net approach for transactional-qos driven web service composition,” International Journal of Web and Grid Services, vol. 7, no. 1, pp. 91–115, 2011.
[3] J. El Hadad, M. Manouvrier, and M. Rukoz, “Tqos: Transactional and qos-aware selection algorithm for automatic web service composition,” Services Computing, IEEE Transactions on, vol. 3, no. 1, pp. 73–85, 2010.
[4] Y. Cardinale, J. El Haddad, M. Manouvrier, and M. Rukoz, “Web service composition based on petri nets: Review and contribution,” in Resource Discovery, ser. Lecture Notes in Computer Science, Z. Lacroix, E. Ruckhaus, and M.-E. Vidal, Eds. Springer Berlin Heidelberg, 2013, vol. 8194, pp. 83–122. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-45263-5_5
[5] K. Lawrence and C. Kaler, “Web services security: Soap message security 1.1 oasis standard specification,” 2006. [Online]. Available: http://docs.oasis-open.org/wss/v1.1
[6] ——, “Web services security: X.509 certificate token profile 1.1 oasis standard specification.” [Online]. Available: http://docs.oasis-open.org/ wss/v1.1
[7] Z. Qi, “Secure digital certificate design based on the public key cryptography algorithm,” 2013.
[8] B. Carminati, E. Ferrari, and P. Hung, “Security conscious web service composition with semantic web support,” 23rd International IEEE Conference on Data Engineering, 2007.