INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS MONTERREY

PROGRAMA DE GRADUADOS EN ELECTRÓNICA, COMPUTACIÓN, INFORMACIÓNY COMUNICACIONES

UN ANÁLISIS EXPERIMENTAL DEL DESEMPEÑO DE WEB SERVICES EMPLEANDO FIRMAS DIGITALES COMO MECANISMO DE SEGURIDAD

TESIS

PRESENTADA COMO REQUISITO PARCIAL PARA OBTENER EL GRADO ACADEMICO DE:

MAESTRÍA EN CIENCIAS EN TECNOLOGÍA INFORMÁTICA

POR:

OTHONIEL ISAAC POOL COUOH

MONTERREY , N.L. NOVIEMBRE 2004

INSTITUTO TECNOLÓGICO DE ESTUDIOS SUPERIORES DE MONTERREY

DIVISIÓN DE ELECTRÓNICA, COMPUTACIÓN, INFORMACIÓN Y COMUNICACIONES

PROGRAMAS DE GRADUADOS EN ELECTRÓNICA, COMPUTACIÓN, INFORMACIÓN Y COMUNICACIONES

Los miembros del comité de tesis recomendamos que la presente tesis del Ing. Othoniel Isaac Pool Couoh sea aceptada como requisito parcial para obtener el grado académico de Maestro en Ciencias en Tecnología Informática.

Comité de tesis:

_____

Alejandro Parra Briones, MC.
Asesor

_____

José Raúl Pérez Cázares, PhD.
Sinodal

_____

Guillermo Jiménez Pérez, PhD
Sinodal

_____

David Alejandro Garza Salazar, PhD.
Director del Programa de Graduados en Electrónica, Computación, Información y Comunicaciones.
Noviembre de 2004

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS MONTERREY

PROGRAMA DE GRADUADOS EN ELECTRÓNICA, COMPUTACIÓN, INFORMACIÓNY COMUNICACIONES

UN ANÁLISIS EXPERIMENTAL DEL DESEMPEÑO DE WEB SERVICES EMPLEANDO FIRMAS DIGITALES COMO MECANISMO DE SEGURIDAD

TESIS

PRESENTADA COMO REQUISITO PARCIAL PARA OBTENER EL GRADO ACADEMICO DE:

MAESTRÍA EN CIENCIAS EN TECNOLOGÍA INFORMÁTICA

POR:

OTHONIEL ISAAC POOL COUOH

MONTERREY , N.L. NOVIEMBRE 2004

Dedicatoria

A mi Dios, que en todo momento ha estado conmigo demostrándome lo mucho que me quiere. Sin Él ningún logro tiene valor.

A mis padres, por enseñarme con su ejemplo el camino de la perseverancia, por todo el amor que siempre me han dado y porque siempre me han apoyado en las decisiones que he tomado.

A mi hermana, porque ha sido no sólo una hermana, ha sido una amiga en quien confiar.

A mis sobrinos, por su ternura y amor que han tenido hacia conmigo.

A Miriam, por lo especial que es en mi vida, por el amor que me da y porque sin ella hubiera sido muy difícil el estar lejos de mi familia durante la maestría.

A mis familiares que siempre han estado al pendiente de mis cosas y me han brindado su cariño.

A mis amigos, tanto los que son de toda la vida como los que conocí en estos últimos dos años, porque siempre me impulsaron a seguir adelante y nunca decaer.

Índice de Figuras

Agradecimientos

A mi asesor, M.C. Alejandro Parra Briones, por guiarme y apoyarme en el desarrollo de este proyecto.

A mis sinodales, el Dr. Rául Pérez y el Dr. Guillermo Jiménez, por sus comentarios y opiniones que ayudaron a enriquecer este trabajo.

A mis familiares y amigos por sustentarme no sólo durante este proyecto sino en todo tiempo.

Resumen

La Internet es considerada una de la redes más inseguras porque su diseño original no tiene mecanismos robustos de seguridad, aún así, todos los días se siguen realizando transacciones sin controles de seguridad o si se emplean controles la mayoría de las veces no son suficientes para proporcionar los requerimientos básicos de seguridad.

Con el surgimiento de nuevas tecnologías se hacen necesarias nuevas especificaciones de seguridad. Una de estas tecnologías recientes son los Web Services que permiten proveer servicios en la Web a través de protocolos sumamente interoperables. Para proporcionar servicios a través de Internet se necesitan cubrir los requerimientos de seguridad y de desempeño entre otros. La interoperabilidad de los Web Services trae consigo problemas de seguridad. Al ser los Web Services otra tecnología de Internet, los controles existentes para aplicaciones Web se pueden emplear sin problemas para los Web Services. Pero es importante observar que ninguno de estos controles reside en la capa de aplicación. Por la necesidad de contar con especificaciones de seguridad a nivel de aplicación para Web Services surge entre otras la especificación de Firmas XML. Las Firmas XML cubren algunos de los requerimientos más importantes para la seguridad en la transmisión de mensajes como son la integridad, la autenticación del mensaje y/o la autenticación del firmante.

Hasta el momento no se tiene conocimiento de la existencia de estudios del impacto que se produce en el desempeño cuando los Web Services emplean Firmas XML, dicho desempeño está comprendido principalmente por el tiempo de respuesta de un Web Service. Es necesario saber si los Web Services con Firmas XML cubren los dos requerimientos mencionados anteriormente.

Esta tesis pretende ser un punto de partida para el análisis del desempeño de los Web Services con Firmas XML. El presente trabajo es un análisis experimental del desempeño de Web Services que emplean Firmas XML como mecanismo de seguridad. En él se describe el diseño de un conjunto de Web Services desarrollados con distintas implementaciones y con distintas librerías de Firmas XML disponibles actualmente, al igual que el diseño de series de pruebas con diferentes especificaciones para obtener el comportamiento de desempeño de los Web Services implantados. A través de los resultados de los experimentos se pudieron obtener conclusiones importantes acerca del comportamiento del desempeño en diferentes circunstancias.

Tabla de Contenido

Índice de Listados

Índice de Tablas

Capítulo 1

Introducción

La considerable expansión que ha tenido Internet en los últimos años, ha impulsado el uso de tecnologías que permiten proveer servicios a través de Internet. La reducción del tiempo y los recursos empleados en los procesos siempre han sido prioridades básicas en los negocios debido a su impacto económico. Idealmente, se desea invertir sólo en lo suficiente y no más, destinando más recursos en las partes donde se generan mayores beneficios. Para lograr esto, se hace necesario contar con sistemas robustos que permitan mantener las propiedades del servicio según las exigencias establecidas [43].

Otra tendencia de nuestros tiempos es la globalización, el libre intercambio de bienes y servicios a nivel global. En esta forma de comercio, las telecomunicaciones juegan un papel preponderante. Al principio, el comercio electrónico sólo se daba en una relación negocio-cliente, actualmente existen otras relaciones como negocio-negocio y gobierno-ciudadanos que requieren de un mayor grado de automatización y en algunos casos deben operar con una mínima intervención humana [44].

Para poder proporcionar estos servicios se necesitan de sistemas de información tan eficientes como los clientes de dichos servicios lo demanden. En algunos casos se establecen Acuerdos de Nivel de Servicios (SLAs, Service Level Agreements) entre los proveedores y los clientes sobre la Calidad del Servicio (QoS, Quality of Service). Estos SLAs son contratos legales que establecen límites en las métricas de la Calidad del Servicio [16].

Otro aspecto a considerar no sólo en las aplicaciones empresariales, sino en cualquier sistema informático es la seguridad de la información. En un ambiente distribuido la necesidad de proteger la información es mucho más grande, por ello, las aplicaciones para proporcionar servicios deben satisfacer las políticas de seguridad impuestas por los clientes que requieren dichas aplicaciones.

Los Web Services son un mecanismo reciente que permite satisfacer adecuadamente las necesidades de negocio anteriormente señalados. Harvey Deitel afirma que “la tecnología de Web Services -la cual representa la próxima etapa en computación distribuida- afectará profundamente a las organizaciones en el 2002 y después”. Igualmente menciona que “las compañías están implementando Web Services para facilitar una amplia variedad de procesos de negocios, tales como la integración de aplicaciones y transacciones negocio a negocio” [1].

Originalmente, las especificaciones para Web Services no proporcionaban mecanismos para proveer seguridad en los datos. En la Recomendación de SOAP Versión 1.2 [6], el grupo de redactores indicaba que “SOAP no menciona…aspectos tales como…la transferencia confiable de datos, atravesamiento de cortafuegos, etc.”

Debido a la demanda de seguridad por parte de todos los sectores interesados en el uso de Web Services, han surgido especificaciones que permiten proporcionar mecanismos de seguridad preservando la interoperabilidad entre dichos programas [7,8, 9, 10]. Estas especificaciones están basadas en métodos y técnicas que han comprobado su efectividad desde hace muchos años. Hablando específicamente, se trata de la criptografía y la encriptación de mensajes. Estas técnicas fueron adaptadas a la tecnología de Web Services para implementar la seguridad en la capa de aplicación.

Una de estas tecnologías son las Firmas Digitales aplicables sobre los mensajes que intercambian los Web Services. Las Firmas Digitales cubren algunos de los requerimientos más importantes para la seguridad en la transmisión de mensajes como son la integridad, la autenticación del mensaje y/o la autenticación del firmante.

Una vez teniendo estas especificaciones de seguridad para Web Services surge otra consideración, ¿qué tanto afectan los mecanismos de seguridad al desempeño de los Web Services?, Joanne Martin et al. menciona que entre los retos técnicos para los desarrolladores de tecnologías para Web Services se encuentra el aspecto de desempeño, indica que “el cifrado y descifrado también incrementan el overhead. Estos aspectos de desempeño serán tratados conforme las tecnologías maduren…” [22].

Niels Ferguson y Bruce Schneier en la sección “Los demonios del desempeño” exponen su filosofía personal de que: “queremos que el sistema sea tan eficiente como sea posible, pero no a expensas de la seguridad” [27].

Así también, Freeman y Miller afirman que “los controles criptográficos son muy costosos para sistemas de desempeño crítico y de tiempo real…La cantidad de tiempo para ejecutar controles criptográficos claramente depende del esquema de seguridad escogido” [25].

Es por ello que en el presente trabajo se analiza el impacto que produce en el desempeño uno de estos mecanismos de seguridad para Web Services: las Firmas Digitales XML. Para llegar a este fin se realizaron una serie de pruebas sobre diferentes implantaciones de Web Services con Firmas XML.

En base a este estudio se presentarán conclusiones determinantes sobre la relación entre el desempeño y la seguridad de los Web Services.

1.1 Objetivos

Realizar un análisis experimental del desempeño de Web Services empleando Firmas Digitales XML como mecanismo de seguridad.

Comparar el comportamiento del desempeño de los Web Services, implantando las Firmas Digitales XML, para conocer las condiciones bajo las cuales se desempeñan mejor.

Realizar una investigación que sirva de referencia y pueda ayudar a determinar el costo en desempeño que implica implementar Firmas Digitales XML en los Web Services.

1.2 Alcance

El análisis del desempeño se realizará en base al tiempo que toma realizar una llamada a un Web Service y recibir la respuesta. Cuando el mensaje que contiene la invocación al servicio está firmado digitalmente, lleva una carga adicional de procesamiento y transmisión. Eso conlleva a un retraso que se verá reflejado en el tiempo de respuesta del Web Service. El análisis experimental se hará sobre la variación del tiempo de respuesta con mensajes del mismo tipo, firmados y no firmados digitalmente.

Al estar usando la tecnología de Web Services, se emplearán Firmas Digitales XML en los Mensajes SOAP con el estándar proporcionado por la W3C.

La implantación se pretende realizar en computadoras de un solo procesador, ejecutando el sistema operativo Windows XP con el Service Pack 1 instalado. Los procesadores serán de arquitectura Intel x86.

Las pruebas se realizarán primeramente con el cliente y el servidor en el mismo host, y luego se repetirán las pruebas con el cliente y el servidor en diferentes hosts y unidos a través de un switch en una pequeña LAN que tendrá una velocidad de 100 Mbps.

La realización de los experimentos seguirá un esquema muy similar al trabajo presentado por Dan Davis y Manish Parashar [17]. La diferencia principal con dicho trabajo será el empleo de mecanismos de seguridad, en nuestro caso particular las Firmas XML.

1.3 Estructura de la Tesis

El contenido del presente documento sigue la siguiente estructura:

En el Capítulo 1 se presenta una introducción que incluye el contexto, una breve justificación, los objetivos planteados y el alcance del trabajo realizado.

El Capítulo 2 describe los conceptos que fueron necesarios para realizar esta Tesis. Entre estos conceptos se encuentran los Web Services y sus tecnologías relacionadas, el desempeño de Web Services y trabajos relacionados con este tópico; de igual forma, los conceptos relacionados con seguridad computacional, principalmente enfocados a la criptografía. Una vez presentados estos temas, se puede pasar a la seguridad para Web Services. Como es parte esencial del presente trabajo, la seguridad para Web Services con Firmas XML es explicada con mayor detalle. Al final del capítulo se describen las implementaciones SOAP y las librerías de Firmas XML empleadas para las implantaciones realizadas en este trabajo.

El Diseño Experimental es descrito en el Capítulo 3, en dicho capítulo se describe el diseño de los programas, el diseño de las implantaciones y el diseño de las pruebas. Se presenta el esquema de los Web Services a un nivel general primeramente, y luego a un nivel particular dependiendo de las implementaciones SOAP y librerías de Firmas XML empleadas. Después de especificar el diseño de los Web Services se pasa a la descripción del diseño de la implantación de las Firmas XML sobre los Web Services. Una

vez teniendo estas especificaciones se nombra a cada uno de los diferentes Web Services implementados. Al final del capítulo se hace una descripción de las pruebas realizadas sobre las implantaciones de Web Services.

Los resultados de las pruebas hechas sobre los diferentes Web Services implantados se detallan en el Capítulo 4. En base a dichos resultados se realiza un análisis de los resultados, donde se explica el comportamiento obtenido de los resultados de las mediciones.

Por último en el Capítulo 5 se presentan las conclusiones y el trabajo a futuro relacionado con el presente trabajo.

Capítulo 3

Diseño experimental

3.1 Introducción

En el diseño experimental se llevaron a la práctica los conceptos mencionados en el capítulo anterior. Todo el desarrollo de las pruebas giró sobre el desempeño de los Web Services. Para lograr los objetivos propuestos del presente trabajo, se realizaron dos conjuntos de pruebas, estos conjuntos de pruebas tuvieron una sola diferencia: el uso de Firmas XML para los Mensajes SOAP. En el primer conjunto se realizaron pruebas sobre el desempeño de los Web Services sin Firmas XML. En el segundo conjunto se implantaron los Web Services con Firmas XML y su correspondiente verificación. De aquí en adelante para fines prácticos los Mensajes SOAP serán nombrados simplemente como Mensajes. De igual forma ocasionalmente se abreviará Tiempo de Respuesta como T.R. A continuación se realizará una descripción detallada de dichos diseños.

3.2 Diseño de los Web Services

En esta sección se describirá el diseño de los Web Services desarrollados para el presente trabajo. Existe un diseño general con funcionalidades en común para cada uno de los Web Services hechos, sin importar la implementación empleada, y sin importar si tienen incorporado el mecanismo de Firmas XML o no. También el diseño sufre algunas variaciones para adaptarse a la arquitectura de la implementación usada. En el presente trabajo se empleará Apache Axis y webMethods Glue como las implementaciones para el desarrollo de los Web Services.

3.2.1 Diseño general de los Web Services

Cada uno de los Web Services desarrollados soporta las mismas funcionalidades y en todos los casos se emplearon parsers del tipo DOM. Se hizo de esta forma para poder realizar posteriormente las mediciones sobre los mismos procesos bajo condiciones equitativas. Se pretendió que las variaciones fueran solamente por la implementación de las Firmas XML y de esta forma se pudieran analizar dichas variaciones.

Se desarrolló el programa servidor y el programa cliente. El programa servidor como su nombre lo indica, proporciona los servicios de forma remota. Estos servicios no son más que los métodos implementados en el lado del servidor. El programa cliente es el que invoca de forma remota los servicios que proporciona el servidor.

Los métodos que se implementaron fueron los siguientes:

- void setSize(int tamano) - Establece el tamaño del arreglo de enteros y del String devueltos por getString() y getIntegers().
- String getString() - Devuelve un String del tamaño especificado. El String devuelto tendrá solamente caracteres ‘0’.

- int[] getIntegers() - Devuelve un arreglo de enteros del tamaño especificado. Los valores de los enteros irán del *0* al (*tamaño del arreglo - 1*).

El método setSize sólo sirve para asignar el tamaño del String o del arreglo de enteros que será devuelto con getString y getIntegers respectivamente, por lo que no se realizarán mediciones sobre este método.

Se seleccionó al método getString y al método getIntegers porque devuelven tipos de datos representativos, de esta forma se puede analizar el comportamiento según el tipo de datos devuelto y el tamaño de la respuesta.

El diseño de los Web Services sigue el mismo esquema de la arquitectura cliente/servidor. El cliente realiza una petición al servidor, en este caso un método con sus respectivos parámetros si los requiere. El servidor procesa la petición y devuelve una respuesta al cliente (Figura 12).

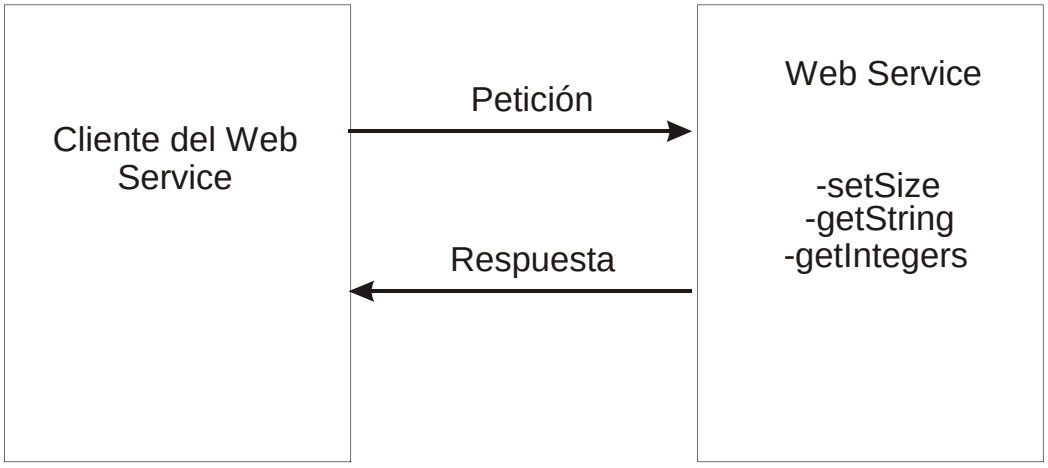

Figura 12. Diseño de los Web Services

Nótese que una particularidad de este esquema cliente/servidor es la conversión previa que debe realizarse antes y después de poner la petición y respuesta en “el hilo”, después de todo, los Web Services emplean Mensajes para comunicarse. Como los programas en Java sólo pueden trabajar con métodos nativos, son necesarias estas conversiones. La Figura 13 ilustra este proceso: antes de mandar la invocación del método deseado se deben convertir los objetos Java en XML para la transmisión sobre algún protocolo de transporte, en este caso HTTP; cuando llega al servidor la petición en XML tiene que ser convertida nuevamente a objetos Java. De regreso, la respuesta debe seguir el mismo proceso de conversión.

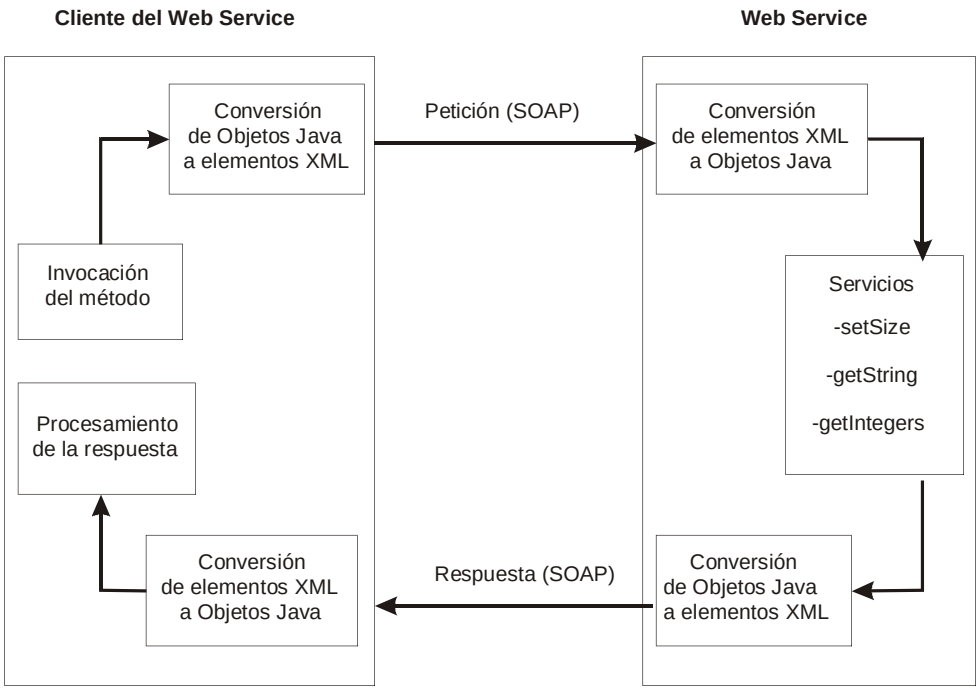

Figura 13. Ilustración de las conversiones entre tipos de datos

3.2.2 Diseño de Web Services con Apache AXIS

La primera implementación para Web Services empleada fue Apache Axis 1.1 trabajando sobre el motor de servlets Tomcat 5.0.19.

Cuando Apache Axis corre sobre un contenedor de servlets, como lo es Tomcat, los servicios están disponibles a través del servlet de Axis. Este servlet contiene todos los Web Services implantados junto con sus respectivos documentos WSDL.

Se emplearon *Servicios RPC* como el estilo de servicio para los Web Services desarrollados. Este estilo es el predeterminado en Axis, y sigue las reglas de SOAP RPC y su codificación. Cada invocación es modelada como un elemento de salida el cual coincide con el nombre de la operación, y también contiene los parámetros para la operación. Axis deserializa el mensaje XML a objetos Java para que sean utilizados por el servicio, y luego serializa los objetos java del servicio a XML para retornarlos al cliente.

3.2.3 Diseño de Web Services con webMethods Glue

Con la implementación para Web Services webMethods Glue se empleó el *modo independiente* de Glue. En *modo independiente*, Glue es esencialmente un servidor de aplicaciones ligero y de alto desempeño. En este modo, Glue usa su propio contenedor de servlets para procesar las peticiones: HTTP, de servlets, y de JSPs, por lo que no requiere alguna otra aplicación de terceros.

Se empleó el estilo RPC para la invocación de los métodos contenidos en el Web Service. Los métodos son especificados en una interfase, y la clase que contendrá los servicios implementa dicha interfase, al mismo estilo que Java RMI. El programa servidor es el encargado de publicar la clase remota como Web Services.

3.3 Diseño de la Implantación de Firmas XML sobre los Web Services

Para poder analizar el comportamiento del desempeño de los Web Services con Firmas XML se aplicaron los siguientes lineamientos para todos los Web Services:

- La generación de Firmas XML (ver **sección 2.6.1.5**) es aplicada a los Mensajes solamente por el servidor.
- La Firma XML se añade a la respuesta que el servicio devuelve a la petición del cliente.
- La verificación de la Firma XML (ver **sección 2.6.1.6**) es aplicada a los Mensajes solamente por el cliente.
- La verificación de la Firma XML se realiza sobre la respuesta que el servicio devuelve al cliente.
- El algoritmo utilizado para la generación de las Firmas XML fue el RSA-SHA1 [7].
- Se empleó el algoritmo de hash (ver **sección 2.5.4**) SHA1 [31] para generar el resumen del Mensaje.
- Se empleó el algoritmo de canonización (ver **sección 2.6.1.2**) Canonical XML 1.0 [34].
- Las Firmas XML implantadas fueron del tipo *enveloped signature* (ver **sección 2.6.1.1)**.

3.4 Diferentes Web Services implantados

La Figura 12 muestra las variaciones en los Web Services implantados. Las diferencias de los Web Services radican en lo siguiente:

- La implementación usada para el desarrollo e implantación de los Web Services.
- Si el Web Service usa o no una Firma XML como mecanismo de seguridad.
- Si el Web Service tiene Firma XML, la librería empleada para realizar el firmado y la verificación de la Firma XML.

Como se puede apreciar en la Figura 14, existen 6 diferentes Web Services implantados y sus diferencias se basan en los puntos mencionados anteriormente. Cada uno de estos Web Services tiene proporciona los servicios descritos en la **sección 3.2.1**.

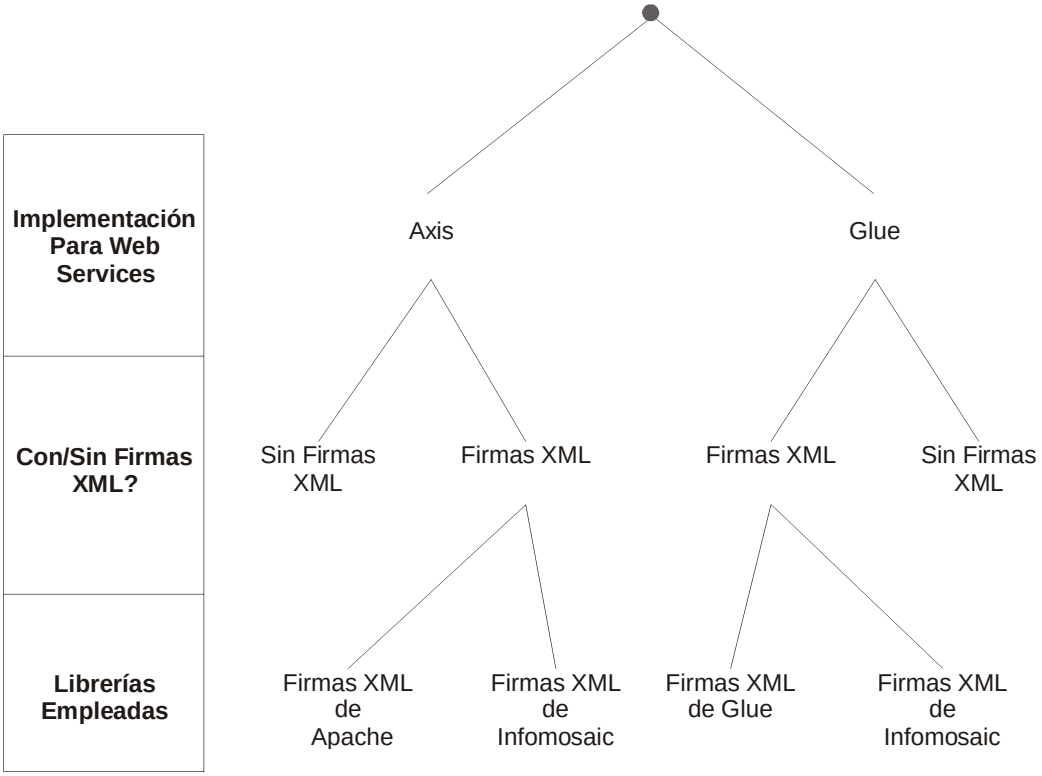

Figura 14. Implementaciones y librerías empleadas para los Web Services

3.5 Diseño de las pruebas

Las pruebas realizadas se aplicaron a cada uno de los diferentes Web Services mencionados en la **sección 3.4**.

Estas pruebas se realizaron con el cliente y el servidor en el mismo host, y con el cliente y el servidor en diferente host.

Para las pruebas en un solo host se empleó la Computadora 1, sus características son las siguientes:

- Laptop Pentium 4 a 2.66 Ghz
- 512 memoria RAM
- Sistema operativo Windows XP Professional Service Pack 1

Para las pruebas con el cliente y el servidor en diferente host, se empleó la Computadora 1 y la Computadora 2. Esta última con las siguientes características:

- Laptop Pentium 4 a 1.7 Ghz
- 256 memoria RAM
- Sistema operativo Windows XP Home Edition Service Pack 1

La Computadora 1 actúa como servidor y la Computadora 2 opera como cliente. Los hosts se conectaron con una pequeña LAN a través de un switch con velocidad de 100Mbs. Sólo estas dos computadoras estuvieron conectadas a la LAN.

Las pruebas consistieron de lo siguiente:

- Medir el tiempo total que toma realizar una invocación y recibir la respuesta del Web Service.
- Aplicar las mediciones a los métodos getString y getIntegers mencionados en la **sección 3.2.1**.
- Para el método getString, se midió el tiempo con longitudes en el String devuelto por el servicio que van desde *0* hasta *102400*.
- Para el método getIntegers, se midió el tiempo con *0, 200, 400, 800, 1600* y *3200* enteros en el arreglo devuelto por el servicio.
- Para cada uno de los métodos se obtuvo el tamaño del Mensaje.

Para calcular el tiempo de ida y vuelta de cada llamada se ejecutaron conjuntos de 1000 llamadas y se midió el tiempo total transcurrido para estas 1000 llamadas, después se dividió este tiempo entre 1000 para saber el tiempo transcurrido por cada llamada. Estas mediciones fueron aplicadas a los métodos implantados con las diferentes longitudes especificadas.

Para el método getString se seleccionaron longitudes de 0 hasta 102400 porque basándose en el trabajo de Dan Davis y Manish Parashar [17] se sabe que la latencia para un Mensaje SOAP que devuelve sólo un String aumenta en base al tiempo de transferencia, por lo tanto se pretende abarcar hasta una longitud considerable.

Se seleccionaron para el método getIntegers longitudes de 0, 200, 400, 800, 1600 y 3200 para el arreglo de enteros basándose en [17], se sabe que el tiempo requerido para devolver la respuesta crece más rápido que la longitud del arreglo de enteros.

Se agruparon los resultados de las mediciones en base a la implementación SOAP empleada para el desarrollo e implantación. De esta forma, las comparaciones y análisis se hicieron sobre los diferentes tipos de Web Services en el grupo. A su vez, dentro de cada grupo se realizó otra agrupación en base al método invocado, para observar el comportamiento del tiempo transcurrido para cada método.

Los grupos para el análisis y presentación de resultados quedaron de la siguiente manera:

1) Grupo de Axis:
    a. Web Service sin Firma XML.
    b. Web Service con Firma XML (implementada con XML Security de Apache).
    c. Web Service con Firma XML (implementada con SecureXML de Infomosaic).

2) Grupo de Glue:
    a. Web Service sin Firma XML.
    b. Web Service con Firma XML (implementada con XML Digital Signatures de Glue).
    c. Web Service con Firma XML (implementada con SecureXML de Infomosaic).

Capítulo 4

Resultados y Análisis

Después de realizar las pruebas descritas en el capítulo anterior, se obtuvieron los resultados de las mediciones hechas a los tiempos de respuesta de los servicios implantados. En este capítulo se presenta la siguiente información: el tamaño de los Mensajes de los métodos implantados, los resultados de las mediciones y al final el análisis de la información obtenida. En la sección Análisis de Resultados se presentan los resultados en forma de gráficas para su comparación y análisis.

4.1 Tamaño de los Mensajes

4.1.1 Tamaño de los Mensajes en el Grupo de Axis

Tabla 3. Tamaños de Mensajes para el método getString() en el grupo de Axis

Tamaño del String	Tamaño del Mensaje (bytes)		
	Sin Firma XML	SecureXML de Infomosaic	XML Security de Apache
0	479	3584	2558
200	700	3805	2779
400	900	4005	2979
800	1300	4405	3379
1600	2100	5205	4179
3200	3700	6805	5779
6400	6900	10005	8979
12800	13300	16405	15379
25600	26100	29205	28179
51200	51700	54805	53779
102400	102900	106005	104979

Tabla 4. Tamaños de Mensajes para el método getIntegers() en el grupo de Axis

Tamaño del Arreglo De enteros	Tamaño del Mensaje (bytes)		
	Sin Firma XML	SecureXML de Infomosaic	XML Security de Apache
0	577	3682	2656
200	4696	7801	6775
400	8896	12001	10975
800	17296	20401	19375
1600	34697	37802	36776
3200	69897	73002	71976

4.1.2 Tamaño de los Mensajes en el Grupo de Glue

Tabla 5. Tamaños de Mensajes para el método getString() en el grupo de Glue

Tamaño del String	Tamaño del Mensaje (bytes)		
	Sin Firma XML	SecureXML de Infomosaic	XML D. Signatures de Glue
0	258	3379	2477
200	466	3587	2685
400	666	3787	2885
800	1066	4187	3285
1600	1866	4987	4085
3200	3466	6587	5685
6400	6666	9787	8885
12800	13066	16187	15285
25600	25866	28987	28085
51200	51466	54587	53685
102400	102666	105787	104885

Tabla 6. Tamaños de Mensajes para el método getIntegers() en el grupo de Glue

Tamaño del Arreglo De enteros	Tamaño del Mensaje (bytes)		
	Sin Firma XML	SecureXML de Infomosaic	XML D. Signatures de Glue
0	292	3405	2614
200	2193	5306	4515
400	4193	7306	6515
800	8193	11306	10515
1600	16794	19907	19116
3200	34394	37507	36716

4.2 Resultados del Grupo de Axis

En esta sección se presentan los resultados de las mediciones hechas con los diferentes tipos de Web Services contenidos en el Grupo de Axis descrito en la **sección 3.5**. Primeramente se presentan los resultados de las mediciones aplicadas al método getString, y después se presentan los resultados del método getIntegers.

4.2.1 Resultados con el cliente y el servidor en el mismo host

Tabla 7. T.R. para el método getString() del grupo de Axis en el mismo host

Tamaño del String	Tiempo de respuesta (milisegundos)		
	Sin Firma XML	SecureXML de Infomosaic	XML Security de Apache
0	3.174	57.521	229.375
200	3.225	57.582	231.125
400	3.234	57.694	233.244
800	3.294	57.994	233.871
1600	3.425	58.581	234.516
3200	3.625	59.893	237.153
6400	3.906	61.886	242.734
12800	4.626	67.725	253.135
25600	6.254	76.213	283.036
51200	10.249	99.647	353.275
102400	13.757	149.014	432.192

Tabla 8. T.R. para el método getIntegers() del grupo de Axis en el mismo host

Tamaño del Arreglo de enteros	Tiempo de respuesta (milisegundos)		
	Sin Firma XML	SecureXML de Infomosaic	XML Security de Apache
0	3.275	57.564	244.405
200	8.783	250.033	949.667
400	17.555	501.478	1982.584
800	29.061	1322.951	7258.159
1600	56.786	4547.417	21551.272
3200	88.135	36382.273	63042.313

4.2.2 Resultados con el cliente y el servidor en diferente host

Tabla 9. T.R. para el método getString() del grupo de Axis en diferente host

Tamaño del String	Tiempo de respuesta (milisegundos)		
	Sin Firma XML	SecureXML de Infomosaic	XML Security de Apache
0	4.086	186.939	366.613
200	4.366	187.259	369.808
400	4.426	187.950	373.003
800	4.557	188.521	373.596
1600	4.647	189.262	374.627
3200	5.257	190.403	378.839
6400	5.748	195.170	387.755
12800	6.940	202.341	404.368
25600	10.015	218.534	452.135
51200	16.391	257.350	543.500
102400	22.544	360.468	654.836

Tabla 10. T.R. para el método getIntegers() del grupo de Axis en diferente host

Tamaño del Arreglo de enteros	Tiempo de respuesta (milisegundos)		
	Sin Firma XML	SecureXML de Infomosaic	XML Security de Apache
0	4.466	194.726	381.546
200	12.198	685.184	1349.784
400	25.587	1314.528	2488.206
800	42.220	3302.552	6571.873
1600	83.159	10152.684	20202.483
3200	132.561	51791.285	74059.836

4.3 Resultados del Grupo de Glue

En esta sección se presentan los resultados de las mediciones hechas con los diferentes tipos de Web Services contenidos en el Grupo de Glue descrito en la **sección 3.5**. Primeramente se presentan los resultados de las mediciones aplicadas al método doNothing, posteriormente se muestran los resultados obtenidos con el método getString, y por último se presentan los resultados del método getIntegers.

4.3.1 Resultados con el cliente y el servidor en el mismo host

Tabla 11. T.R. para el método getString() del grupo de Glue en el mismo host

Tamaño del String	Tiempo de respuesta (milisegundos)		
	Sin Firma XML	SecureXML de Infomosaic	XML D. Signatures de Glue
0	0.951	77.051	38.348
200	1.011	77.772	38.776
400	1.081	78.122	39.286
800	1.202	78.863	39.417
1600	1.482	80.015	39.767
3200	2.003	81.737	40.728
6400	3.085	85.573	44.424
12800	5.398	96.379	47.669
25600	11.446	121.895	57.993
51200	32.707	189.492	88.988
102400	95.287	379.516	183.844

Tabla 12. T.R. para el método getIntegers() del grupo de Glue en el mismo host

Tamaño del Arreglo de enteros	Tiempo de respuesta (milisegundos)		
	Sin Firma XML	SecureXML de Infomosaic	XML D. Signatures de Glue
0	1.502	78.513	45.245
200	3.926	220.107	520.805
400	7.431	406.735	1402.098
800	14.952	909.258	4626.560
1600	42.872	3087.750	26770.294
3200	91.632	15888.017	109803.830

4.3.2 Resultados con el cliente y el servidor en diferente host

Tabla 13. T.R. para el método getString() del grupo de Glue en diferente host

Tamaño del String	Tiempo de respuesta (milisegundos)		
	Sin Firma XML	SecureXML de Infomosaic	XML D. Signatures de Glue
0	1.172	115.526	42.951
200	1.301	115.916	43.242
400	1.422	116.427	43.262
800	1.732	117.789	43.503
1600	2.223	118.991	44.064
3200	2.904	121.935	45.556
6400	4.406	127.263	48.410
12800	7.921	138.218	54.138
25600	16.694	165.748	72.915
51200	46.387	254.246	114.074
102400	129.085	524.765	209.571

Tabla 14. T.R. para el método getIntegers() del grupo de Glue en diferente host

Tamaño del Arreglo de enteros	Tiempo de respuesta (milisegundos)		
	Sin Firma XML	SecureXML de Infomosaic	XML D. Signatures de Glue
0	1.952	117.639	52.876
200	5.789	329.724	798.578
400	11.096	595.616	1952.888
800	22.408	1294.747	6238.892
1600	61.588	3398.543	30597.500
3200	127.293	15933.992	125273.153

4.4 Análisis de resultados

4.4.1 Análisis del Tamaño de los Mensajes

Inicialmente debe tenerse presente que la representación de datos dentro de un documento XML puede hacerse de diferentes maneras, de esta forma una misma información puede escribirse de diferentes maneras. Basta con tener diferentes nombres en los elementos o atributos de dos o más documentos XML con la misma información para que sus tamaños sean diferentes. En el caso de los Mensajes, pese a que son documentos XML con reglas más específicas, aún existe cierta flexibilidad en su sintaxis. Ejemplos de esta flexibilidad son: la ubicación de los Espacios de Nombres, que pueden estar en el Sobre, en el Cuerpo o en los elementos; y la utilización de Espacios de Nombres, atributos y comentarios opcionales. Por ello, el tamaño de los Mensajes se encuentra fuertemente vinculado con la implementación utilizada y cómo esta implementación escribe los datos en un Mensaje.

Por otro lado, al aplicar una Firma XML a un Mensaje, la librería empleada para tal acción tiene sus propios métodos para el firmado de dicho mensaje. Al igual que con los documentos XML, la sintaxis de las Firmas XML es flexible y por lo tanto un documento XML puede tener diferentes representaciones de Firma XML. El **Apéndice B** contiene

ejemplos de Mensajes mandados por el servidor y recibidos por el cliente donde se tiene la respuesta al método remoto getIntegers() el cual devuelve un arreglo de enteros, para estos Mensajes previamente se había establecido el tamaño del arreglo en 5 para fines prácticos. Se puede ver en los Mensajes del **Apéndice B** las diferencias existentes entre las distintas implementaciones SOAP y las distintas librerías de Firmas XML que afectan directamente al tamaño de los Mensajes

Pueden apreciarse diferencias en los Mensajes tales como el prefijo para el Sobre, que en Axis es *soapenv* y en Glue es simplemente *soap*. También, el nombre de los elementos de la respuesta a la petición, en Axis se nombra a los elementos como *item*, mientras que en Glue se nombra como *i*. Otras diferencias son que la librería SecureXML de Infomosaic explícitamente especifica que el Mensaje tiene una Firma XML del tipo *enveloped signature* y la implementación de Firmas XML de Glue define el método de canonización dentro de las Transformaciones, entre otras.

Después de analizar los Mensajes se puede apreciar que el tamaño de todos los elementos XML generados durante el firmado XML, contenidos dentro del elemento *Signature*, no varía en función del tamaño o número de elementos XML contenidos en el Mensaje, el tamaño de la Firma XML se puede considerar constante.

Por ejemplo, los dos únicos elementos XML que varían en Firmas XML de diferentes Mensajes son DigestValue y SignatureValue. Aunque cada Mensaje diferente tenga estos dos valores diferentes, la longitud de estos dos valores es constante, siempre y cuando se empleen los mismos algoritmos Hash y de firmado. La Tabla 15 presenta dos ejemplos de diferencias y coincidencias entre DigestValue y SignatureValue. Los algoritmos utilizados en la implantación se encuentran en la **sección 3.3**.

Tabla 15. Comparación de DigestValue y SignatureValue de dos Mensajes distintos

Elemento de Firma XML	Retorno de getIntegers() con 1 elemento	Retorno de getIntegers() con 1000 elementos
DigestValue	vTZY9x2fr7P5PDeMbdDHnmFAo8w=	KceVVIHBm8yKJHrdpqK1j7aULas=
SignatureValue	iU1ilY6Aq3RTa+3VJaL6P1MfRUK 9w6z4XotQocO7kAAbFpr50hPTcP NknTUUQT8tEmAmVf66SREvIUdkG 4YdfdwEHVZLFuTF9auaQX8QgCpz 8XpyiydzjkfaKS3qXSp0kuHFP9/ YNb34jXidWt/GLG9wN+LacevcXc ohNlH2oVI=	VYTrY73v/4P26VtGsFwwBgFELM7 VqlPK1s7gefuGPH7e+LPVypQvJV X5bhtuU7UMeY1oaHUQ5KJywgxSO 71vPtHpqQHDeRWs+LCk8bbYbek9 mAOvOv7LYh5s6p4PyPE8F9t/kbA /bjekwFY9UyYKjlnd0DvIOnVZQL nwQqZKKIs=

Tanto DigestValue como SignatureValue siempre serán codificados usando base64 [23]. Para obtener valores con longitudes que sean múltiplos de 4 se rellena el resultado con caracteres ‘=’ definidos en la codificación base64 para ese propósito.

Por ello, el incremento en el tamaño del Mensaje firmado depende exclusivamente de la longitud y cantidad de elementos firmados y no de la Firma XML. La respuesta devuelta para el método getString() es un String contenido en un solo elemento XML, mientras que la respuesta para getIntegers() es un arreglo de enteros donde cada entero

tiene su valor contenido en un elemento XML. Las Figuras 15, 16, 17 y 18 muestran los tamaños de las respuestas para los métodos implantados.

En base a los resultados puede apreciarse que para getString() el tamaño de la respuesta se incrementa paralelamente con el tamaño del String devuelto.

Para getIntegers() el tamaño de la respuesta se incrementa en base al tamaño de los elementos XML que representan a cada entero. Con la implementación Axis, un entero de un solo dígito ocupa 19 bytes en el Mensaje, uno de dos dígitos ocupa 20 bytes, uno de tres dígitos 21 bytes y así sucesivamente.

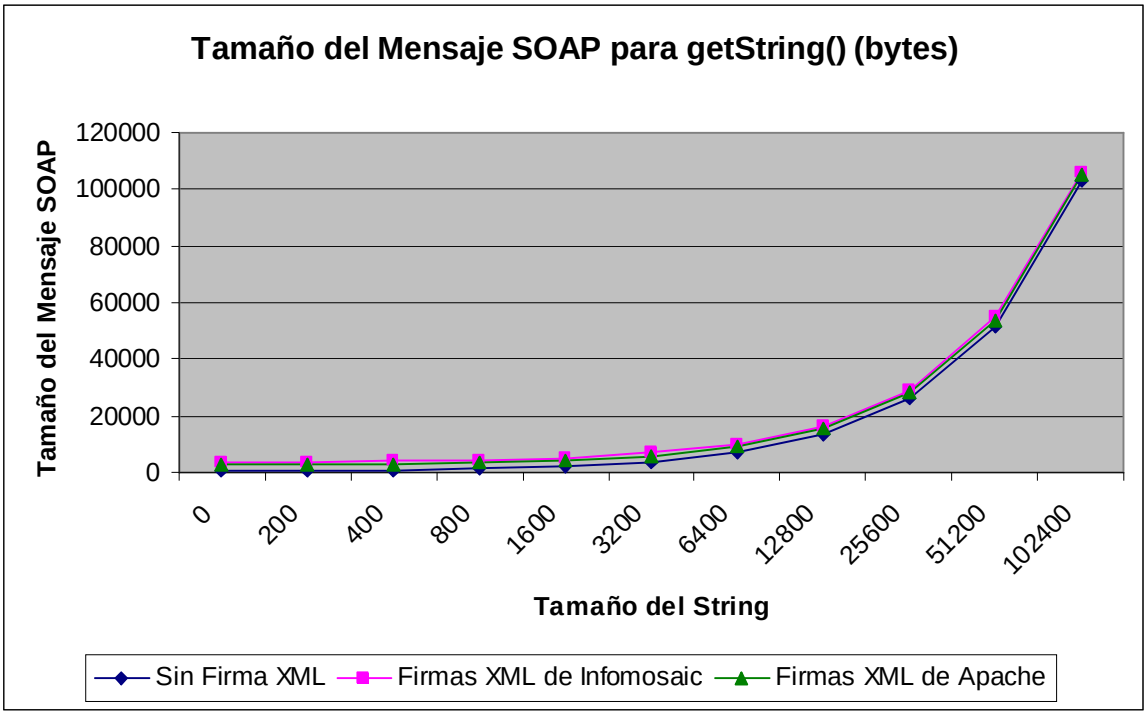

Figura 15. Tamaño del Mensaje para getString() en el grupo de Axis

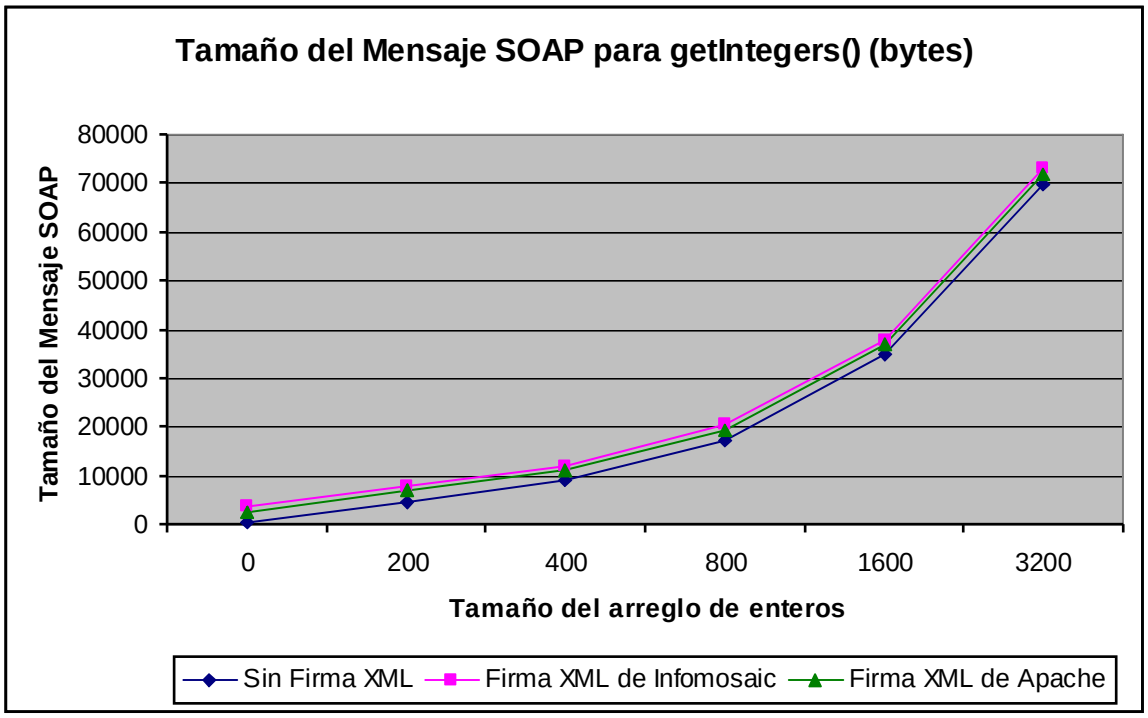

Figura 16. Tamaño del Mensaje para getIntegers() en el grupo de Axis

Con la implementación Glue, los enteros devueltos en el Mensaje ocupan los bytes indicados en la Tabla 16 según su valor. Los valores indicados en la Tabla 16 van del *0* al *9999*. El comportamiento mostrado se explica a continuación.

Se había mencionado en la **sección 3.2.1** que los valores en el arreglo de enteros comienzan con *0* y terminan con (*tamaño del arreglo – 1*), como los enteros son expresados en el Mensaje en forma de elementos XML y estos elementos son sólo texto, si los dígitos de los enteros aumentan el texto también aumenta y en consecuencia el tamaño del Mensaje expresado en bytes. También se debe tener en cuenta la declaración en el Mensaje del tamaño del arreglo que se muestra como en la siguiente línea:

```
<Result arrayType='xsd:int[10]'>
```

En la línea anterior se está especificando un arreglo de enteros de 10 elementos, esta declaración ocupa 1 byte más que una declaración con un arreglo de enteros de 9 elementos por ejemplo, debido a que en texto “9” se representa con 1 byte y “10” con dos bytes. Por lo tanto para saber el tamaño exacto del Mensaje se tienen que sumar los bytes adicionales por la declaración del tamaño del arreglo de enteros.

Rango de valores de los enteros	Bytes necesarios para cada uno de los enteros
0 – 9	8 bytes
10 – 99	9 bytes
100 – 999	10 bytes
1000 – 9999	11 bytes

Tabla 16. Bytes que ocupan los enteros devueltos por getIntegers() según su valor.

Por ejemplo, si se sabe que la respuesta de getIntegers devuelve 1 entero en un Mensaje de 2631 bytes, se puede calcular cuántos bytes tendrá un Mensaje que devuelva 800 enteros con el mismo método remoto. Esto se logra con los cálculos mostrados en la Tabla 17. En esta tabla se indican los bytes que ocupan los enteros según el número de dígitos que tiene dicho entero, se especifica cuáles valores de los enteros son los que cuentan con el número de dígitos indicado. El resultado de la suma puede verificarse en la Tabla 6.

1 entero	Enteros restantes de 1 dígito (del 1 al 9)	Enteros de 2 dígitos (del 10 al 99)	Enteros de 3 dígitos (del 100 al 799)	Bytes adicionales por la declaración del tamaño del arreglo de enteros	Suma
2631 bytes	9*(8 bytes)	90*(9 bytes)	700*(10 bytes)	2	10515

Tabla 17. Cálculos para conocer el tamaño de un Mensaje devuelto por getIntegers().

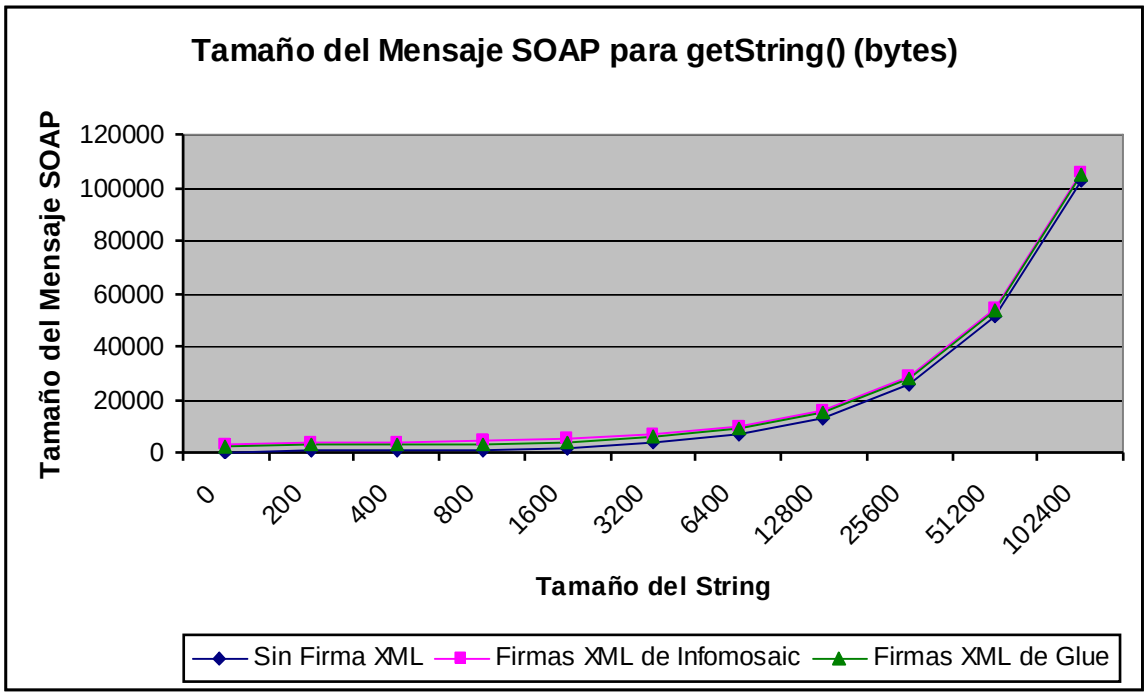

Figura 17. Tamaño del Mensaje para getString() en el grupo de Glue

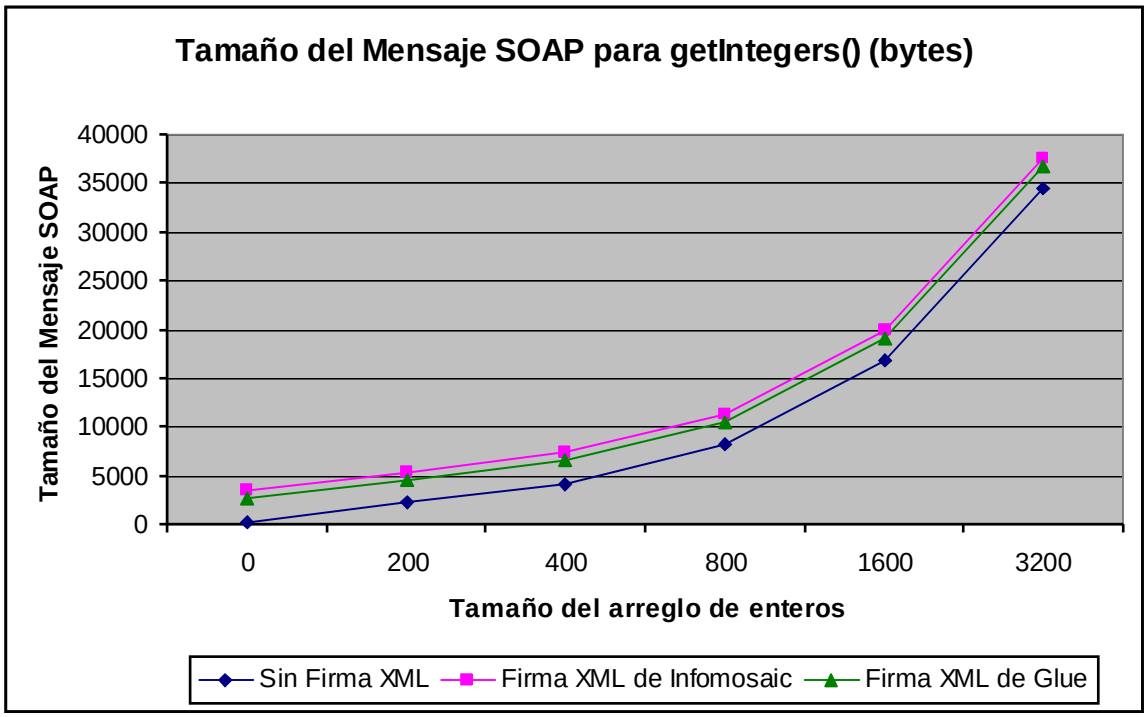

Figura 18. Tamaño del Mensaje para getIntegers() en el grupo de Glue

4.4.2 Análisis de los Tiempos de Respuesta de los métodos implantados

4.4.2.1 El Procesamiento XML afecta el Tiempo de Respuesta de los Web Services

A partir del análisis del tamaño de los Mensajes visto en la sección anterior se puede observar que el tiempo de respuesta no depende exclusivamente del tamaño en bytes del Mensaje. Esta afirmación es aplicable tanto a Mensajes firmados como a los Mensajes no firmados. Por ejemplo, viendo las Tablas 7 y 8 se puede observar que el tiempo de respuesta para getIntegers() con sólo 400 enteros es incluso más largo que el tiempo de respuesta empleado por getString() devolviendo un String de longitud 102400. Viendo las Tablas 3 y 4 se puede comprobar que aun cuando el Mensaje devuelto por getString() es 800% ó más grande que el Mensaje devuelto por getIntegers(), el tiempo de respuesta es menor para getString().

Por otro lado, el tiempo de respuesta se ve afectado por el número de elementos XML que se tienen que procesar. En las implantaciones realizadas se emplearon parsers DOM para el procesamiento XML. Como se sabe el parser es el medio por el cual se puede acceder y actualizar el contenido de documentos XML, en el caso particular de los parsers DOM, estos crean una estructura de árbol en memoria para proporcionar estas capacidades. Para cada nodo de este árbol se tendrá un tiempo de procesamiento que se verá reflejado en el tiempo de respuesta total.

Siguiendo en el mismo sentido, en la **sección 3.2.1** se habían mencionado las conversiones que se requieren realizar entre objetos Java y XML y viceversa con tal de proporcionar la interoperabilidad propia de los Web Services. Al tener un mayor número de elementos XML para convertir, el tiempo de respuesta se sigue incrementando.

Observando los resultados se puede afirmar que el tiempo de respuesta se incrementa en un mayor porcentaje según el número de elementos XML a procesar que por el tamaño de los elementos XML a procesar.

A pesar de que se tienen estos incrementos en el tiempo de respuesta por procesar gran cantidad de elementos XML, una observación a tener en cuenta es el hecho de que sorpresivamente en Mensajes sin Firma XML, la relación (Tiempo de Respuesta) / (Tamaño del arreglo) que nos proporciona un promedio aproximado del tiempo necesario para procesar un entero del arreglo devuelto por getIntegers() se mantiene prácticamente constante e incluso en algunos casos disminuye marginalmente. Este comportamiento puede verse en las Tablas 18 y 19 obtenidas a partir de las Tablas 8 y 12. De igual forma este comportamiento se mantiene con los resultados obtenidos con el cliente y servidor en diferente host.

Tabla 18. Relación entre el T.R. y el Tamaño del arreglo de enteros con Axis

Tamaño del arreglo de enteros	T.R. Sin Firma XML (milisegundos)	Relación: $\frac{Tiempo\_de\_respuesta}{Tamaño\_del\_arreglo}$
200	3.275	0.044
400	8.783	0.044
800	17.555	0.036
1600	29.061	0.035
3200	56.786	0.028

Tabla 19. Relación entre el T.R. y el Tamaño del arreglo de enteros con Glue

Tamaño del arreglo de enteros	T.R. Sin Firma XML (milisegundos)	Relación: $\frac{Tiempo\_de\_respuesta}{Tamaño\_del\_arreglo}$
200	3.926	0.020
400	7.431	0.019
800	14.952	0.019
1600	42.872	0.027
3200	91.632	0.029

4.4.2.2 El Retraso en la Transmisión afecta el Tiempo de Respuesta de los Web Services

Al realizarse las pruebas con el cliente y el servidor en un solo host y también en diferente host se pudieron conocer los porcentajes en los cuales se incrementa el tiempo de respuesta por el retraso de la red. Como se había especificado, la red es una LAN con velocidad de 100Mbps en donde los únicos hosts conectados son el cliente y el servidor. En términos generales, el incremento va de un 14% a un 60% por el retraso en la red cuando se comparan los resultados de las pruebas hechas en un mismo host con los resultados de las pruebas hechas en diferente host.

Salvo algunas excepciones, todas las implantaciones siguen el comportamiento indicado anteriormente. Entre estas excepciones se encuentran algunas implantaciones con la librería SecureXML corriendo sobre Axis que llegan a tener un incremento del

234% respecto de las implantaciones con el cliente y el servidor en el mismo host. Cabe señalar que pese al gran porcentaje de incremento el aumento en tiempo es de unos cuantos milisegundos no mayor a 200.

Como se verá a continuación el retraso en la transmisión a través de la red no es el principal factor que afecta el tiempo de respuesta de los Web Services.

4.4.2.3 El procesamiento de Firmas XML afecta el Tiempo de Respuesta de los Web Services

A través de las Figuras 19 al 22 pueden observarse los tiempos de respuesta para las implantaciones con el cliente y el servidor en el mismo host y las Figuras 23 al 26 proporcionan gráficamente los tiempos de respuesta para las implantaciones con el cliente y el servidor en diferente host.

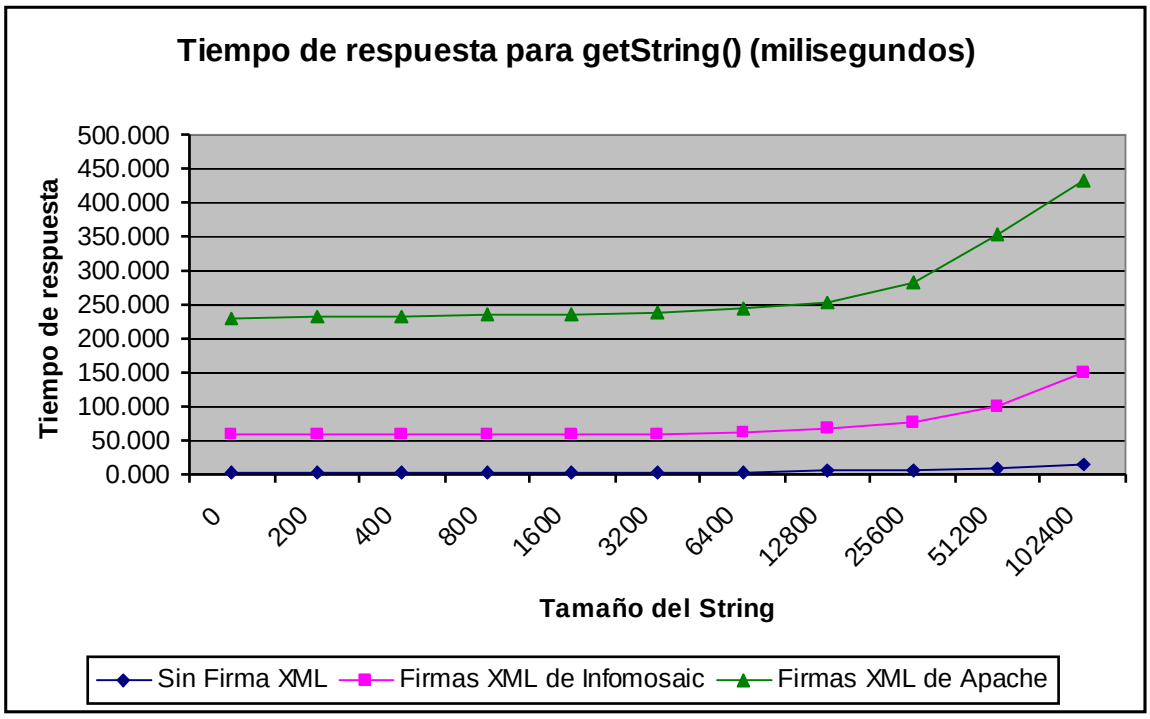

Figura 19. T.R. del grupo de Axis para la invocación getString() en el mismo host

Por observación de las figuras puede afirmarse que del Grupo de Axis la implantación con Firmas XML de Apache es por mucho el de mayor tiempo de respuesta y en consecuencia el menos eficiente. Mientras que en el Grupo de Glue la implantación con Firmas XML de Glue tiene el mayor tiempo de respuesta en las llamadas al método getIntegers() pero escala mejor en las llamadas al método getString() que la implantación con Firmas XML de Infomosaic.

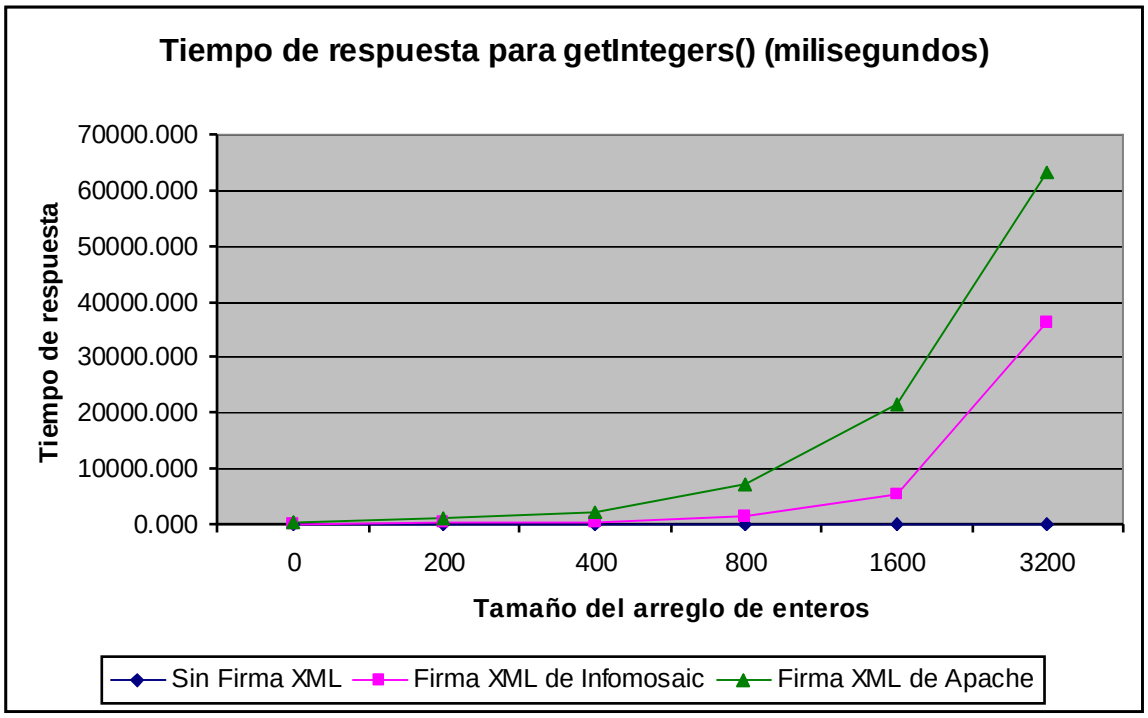

Figura 20. T.R. del grupo de Axis para la invocación getIntegers() en el mismo host

Se puede afirmar también, que la transmisión de Mensajes con la implementación Glue es más eficiente que con la implementación Axis. Pero en cuanto a las librerías de Firmas XML, las que proporciona Apache escalan un poco mejor que las que vienen incluidas en Glue.

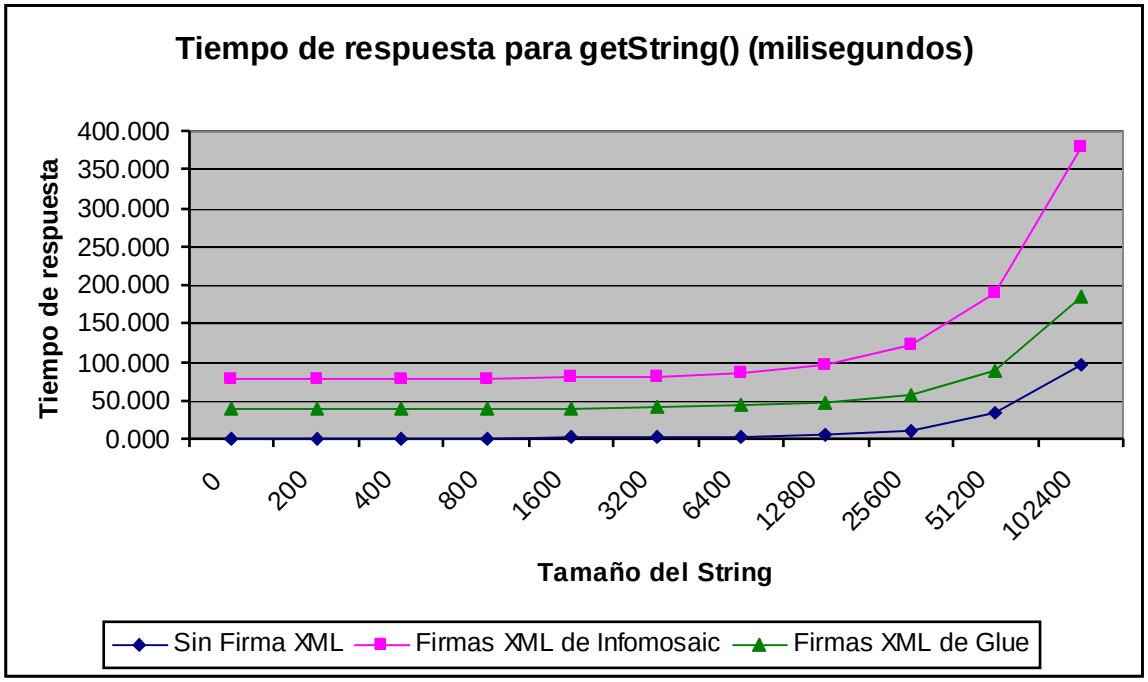

Figura 21. T.R. del grupo de Glue para la invocación getString() en el mismo host

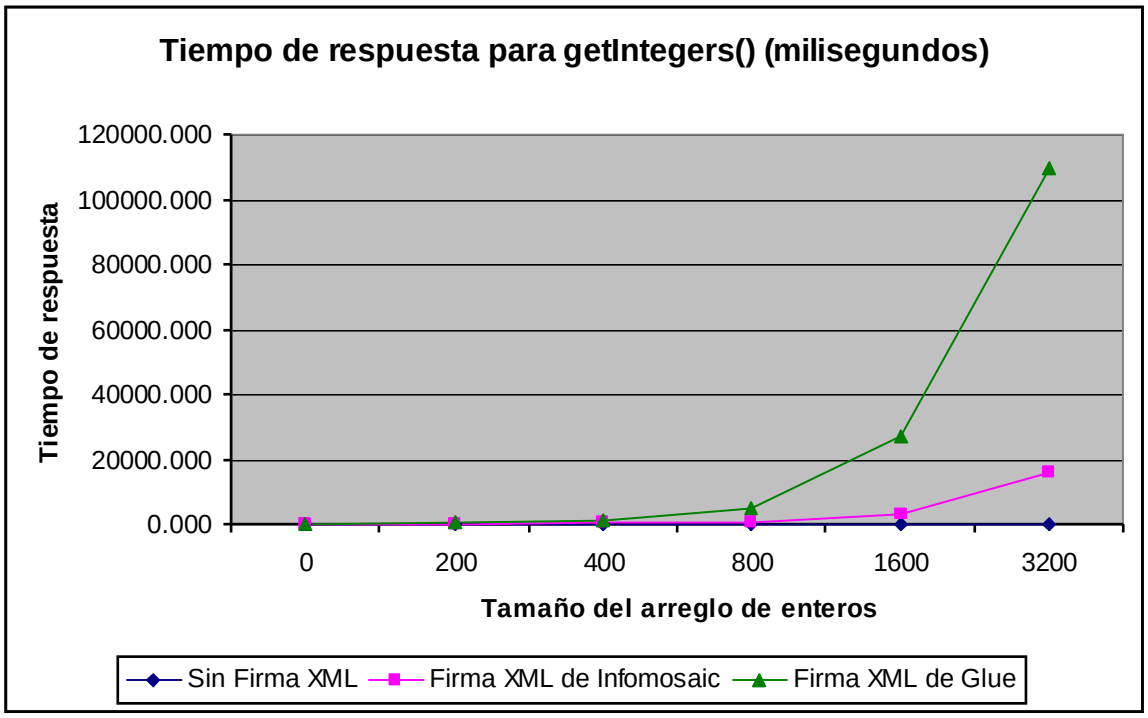

Figura 22. T.R. del grupo de Glue para la invocación getIntegers() en el mismo host

Por otra parte, la librería de Firmas XML proporcionada por Infomosaic tiene un comportamiento estable en todas las pruebas realizadas, pero no por esto proporciona escalabilidad sobre grandes cantidades de elementos XML para firmar.

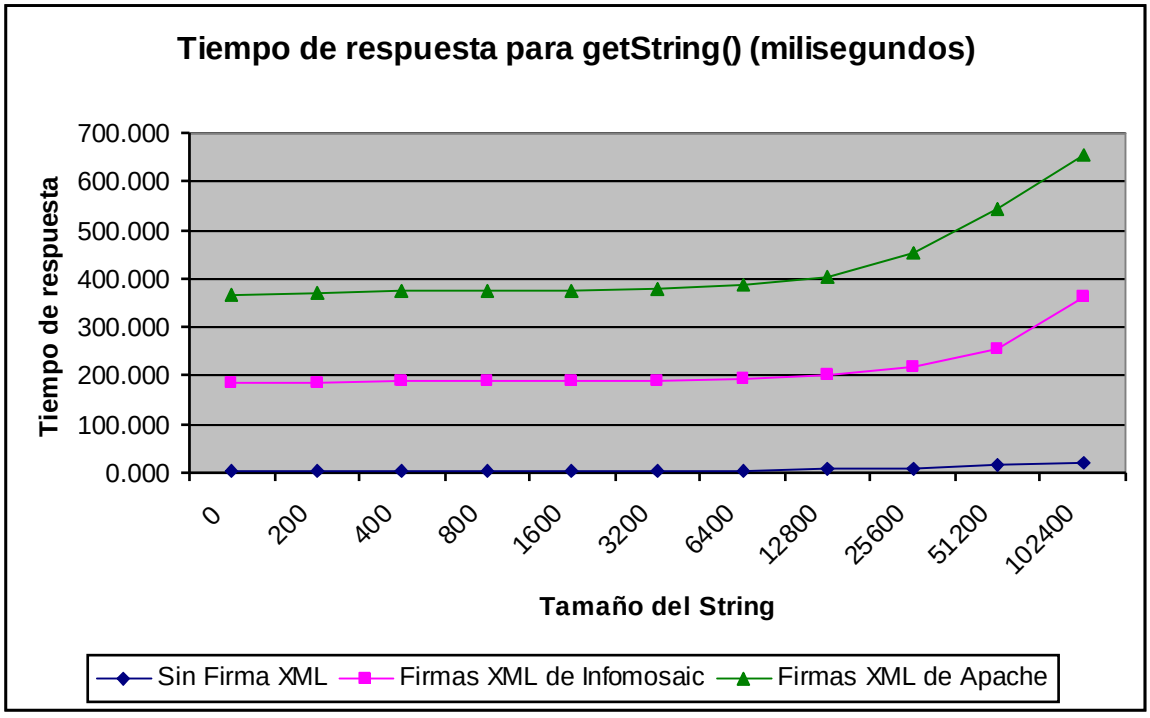

Figura 23. T.R. del grupo de Axis para la invocación getString() en diferente host

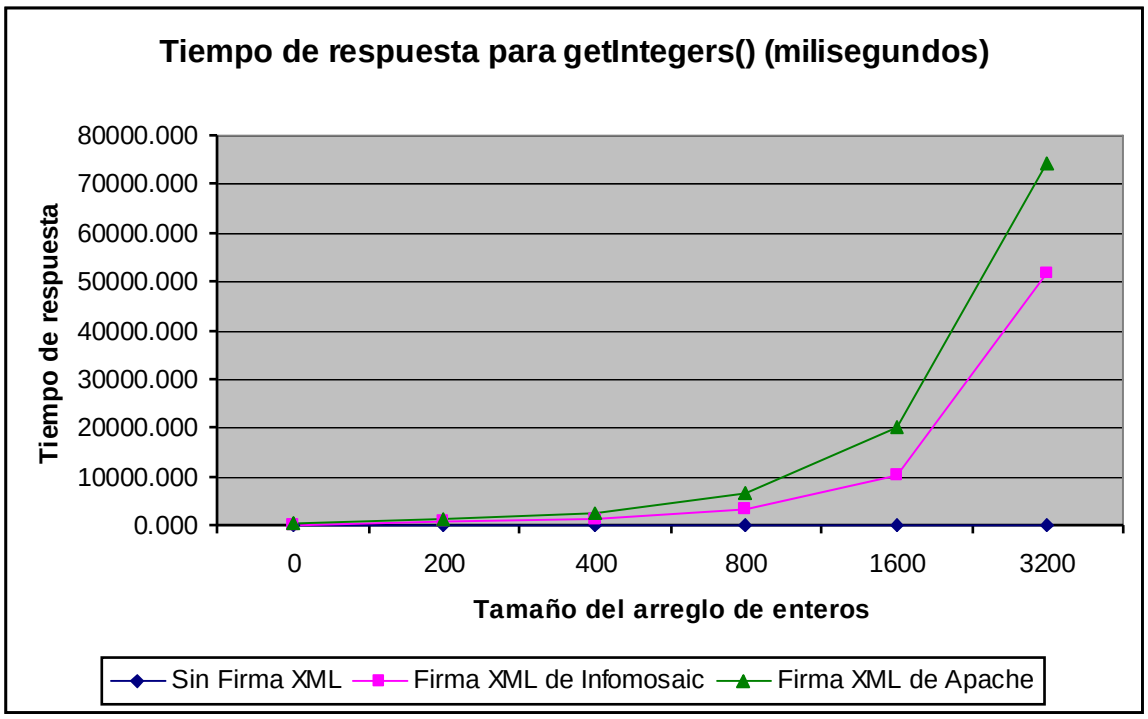

Figura 24. T.R. del grupo de Axis para la invocación getIntegers() en diferente host

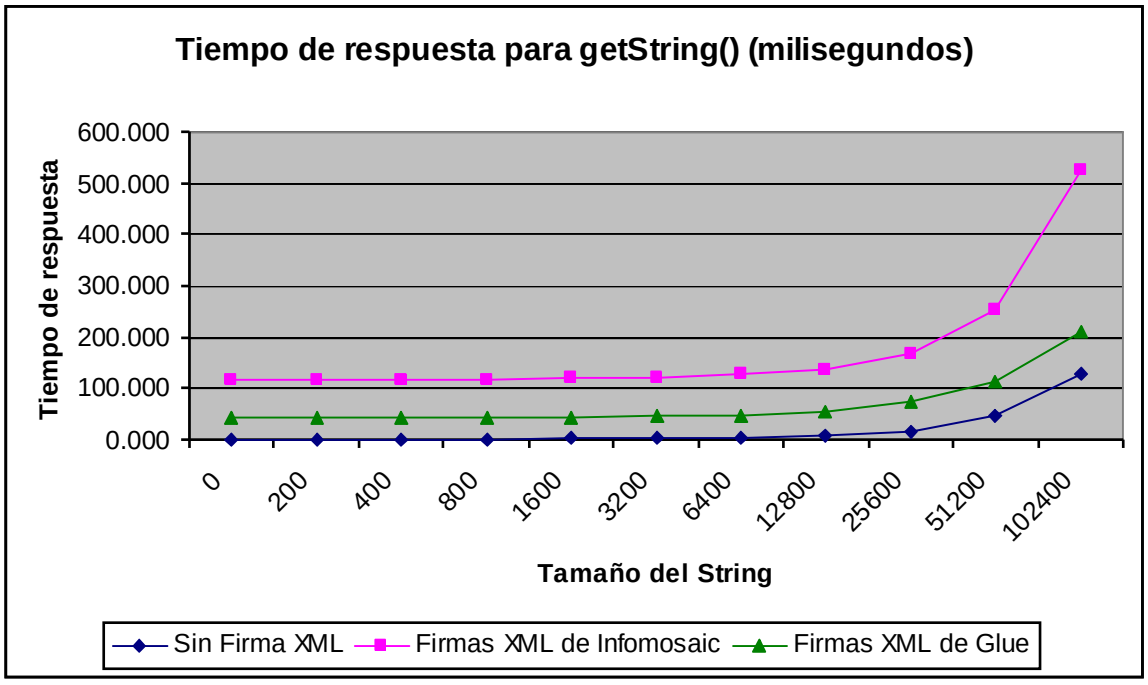

Figura 25. T.R. del grupo de Glue para la invocación getString() en diferente host

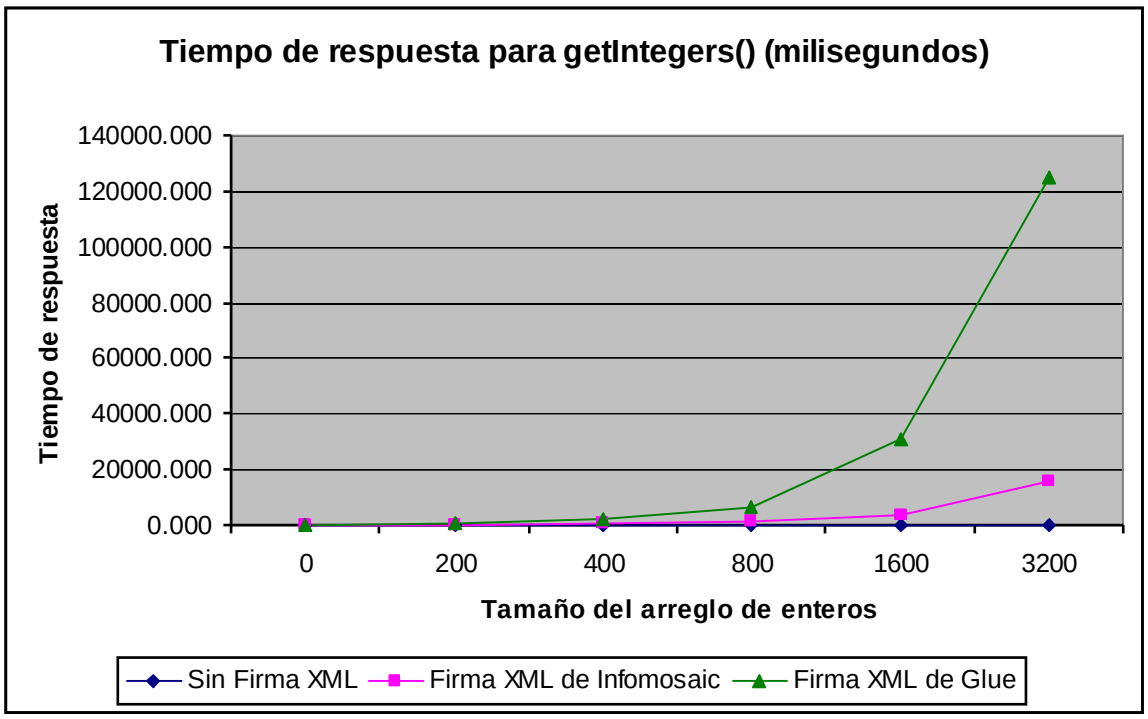

Figura 26. T.R. del grupo de Glue para la invocación getIntegers() en diferente host

En términos generales puede afirmarse que los Web Services con Firmas XML no escalan bien tanto con elementos XML de gran tamaño en el Mensaje como con grandes cantidades de elementos XML en el Mensaje. Aún así, el método getString() proporciona tiempos de respuesta aceptables cuando la respuesta devuelta tiene Firma XML.

Es contrastante observar como el servicio getIntegers() puede llegar a necesitar casi 2 minutos para devolver una respuesta cuando originalmente sólo necesitaba un poco menos de 100 milisegundos.

Con la información presentada hasta este punto se puede afirmar que todo el proceso necesario para la infraestructura de Firmas XML requiere gran cantidad de tiempo de procesamiento. Este es el principal factor que afecta al tiempo de respuesta de los Web Services con Firmas XML como mecanismo de seguridad.

Capítulo 2

Antecedentes

Al paso de los años, los mecanismos de comunicación entre computadoras han cambiado igualmente como han cambiado las necesidades de procesamiento y las aplicaciones. Poco a poco las redes se han ido extendiendo, antes se empleaban terminales para conectarse a las grandes computadoras, ya después se hicieron populares las redes locales y las redes metropolitanas. En nuestros días estas redes se encuentran enlazadas para formar la Internet.

Después que se hizo posible la conexión de computadoras surgió la idea de distribuir las tareas o aplicaciones en computadoras específicas, de tal forma que una máquina proporcionara aplicaciones a las demás; de esta forma apareció la computación distribuida. Más adelante surgieron los protocolos de comunicación que permitieron establecer estándares para la comunicación. Entre los más populares se pueden mencionar CORBA/IIOP (Common Object Request Broker Architecture/Internet Inter-ORB Protocol), DCOM (Distributed Component Object Model) y RMI (Remote Method Invocation) [2].

El auge de Internet lanzó muchos retos computacionales, para empezar era primordial que todas o al menos la mayoría de las tareas que podían realizarse dentro de una red local pudieran llevarse a cabo en Internet, después de todo se puede ver como una sola red. Pero como Internet es independiente de la plataforma, esta característica implica muchos desafíos como por ejemplo la compatibilidad. En el caso de la distribución de las aplicaciones, esto puede provocar que no haya interoperabilidad o que degrade el rendimiento de la aplicación.

El empleo del Web para la utilización de aplicaciones remotas se ha hecho popular gracias a que la interfase para el Web es compatible con todos los ambientes y plataformas. Uno de los primeros intentos de conexión entre aplicaciones Web fue con el uso de marcos (frames) HTML, que permiten utilizar varias aplicaciones al mismo tiempo. Otra técnica consiste en rastrear el contenido de una aplicación Web, extraer la información importante y presentarla como otra versión. La publicación de información consiste en enviar información a un sitio Web como si fuera desde un formato HTML. La publicación de la información se realiza a través de HTTP POST o GET, los dos métodos utilizados en la presentación de información cuando se usa HTTP. Una implementación de esta técnica son los Robots Web [40], programas que atraviesan la estructura de hipertexto de sitios Web recuperando el documento, y recursivamente recuperando todos los documentos que son referenciados.

Las aplicaciones de tipo *salpicadero digital* empleaban todos los métodos mencionados para presentar información de varias fuentes en ventanas o secciones separadas [2].

El problema con los mecanismos mencionados anteriormente radica en la dificultad para darle mantenimiento a la información. Además, un solo error puede propiciar la falla de toda la aplicación.

El protocolo estándar de base XML (Extensible Markup Languaje) proporcionó una solución a la necesidad de interoperabilidad entre aplicaciones Web, esto a través del intercambio de mensajes SOAP (ver **sección 2.2.3**) que son estructurados con XML. Martin Tsenov afirma que "XML es una forma flexible de crear formatos con información común y compartir tanto los formatos como los datos en el Web, las intranets, y otro tipo de redes". Así también indica que "los mensajes SOAP codificados con XML habilitan una comunicación universal entre aplicaciones, servicios y plataformas vía el Internet" [45]. El protocolo XML es indispensable en la tecnología de los Web Services.

2.1 Web Services

Los Web Services, son servicios que se ofrecen a través del Web. Son aplicaciones independientes de la plataforma que pueden ser fácilmente publicadas, localizadas e invocadas mediante protocolos Web estándar, como **XML** (Extensible Markup Languaje), **SOAP** (Simple Object Access Protocol), **UDDI** (Universal Description Discovery and Integration) o **WSDL** (Web Services Definition Service) [3]. En un escenario típico de Web Services, una aplicación de negocios manda una solicitud a un servicio en un URL (Universal Resource Locator) dado usando el protocolo SOAP sobre HTTP (Hypertext Transfer Protocol). El servicio recibe la solicitud, la procesa, y retorna una respuesta. Un ejemplo sencillo de Web Service puede ser la consulta del precio actual de algún producto, y la respuesta dada es el precio de ese producto.

2.2 Tecnologías relacionadas con Web Services

Como se ha mencionado anteriormente existen algunas tecnologías relacionadas con los Web Services, estos son lenguajes o protocolos que permiten comunicar a cada una de las entidades involucradas en la petición y provisión de servicios. El uso de estas tecnologías le brinda a los Web Services su principal característica: la interoperabilidad. En las siguientes secciones se presentan las tecnologías relacionadas con los Web Services.

2.2.1 XML

XML (Extensible Markup Language). Es un lenguaje de marcado parecido al HTML (Hypertext Markup Languaje) en donde las instrucciones pueden ser definidas por etiquetas. Ambos lenguajes son independientes de la plataforma y son un estándar en el desarrollo Web. A diferencia del HTML que fue diseñado para desplegar datos y se enfoca en cómo se ven, el XML fue diseñado para describir los datos y se enfoca en lo que son. En XML las etiquetas no están predefinidas, el programador define sus propias etiquetas, por lo que XML es extensible [3].

XML se derivó de SGML (Standard Generalizad Markup Languaje), un estándar para organizar y etiquetar elementos de un documento. SGML fue desarrollado y estandarizado por la Organización Internacional de Estándares (ISO, International Organization for Standarizatión) en 1986 [4].

La recomendación XML especifica las siguientes metas:

- XML debe ser usable sobre Internet.
- XML debe soportar una variedad de tipos de aplicaciones.
- XML y SGML deben ser compatibles.
- Los programas para procesar documentos XML deben ser fáciles de escribir. Específicamente el comité quiso hacer posible para los desarrolladores crear programas útiles que no dependieran de un Documento de Definición de Tipos (DTD, Document Type Definition).
- Por compatibilidad entre documentos XML, características opcionales en XML deben ser mantenidos al mínimo, idealmente cero.
- Un documento XML debe ser legible por humanos y claro.
- El diseño XML debe ser de lectura rápida para proporcionar formatos de datos, abiertos, no-propietarios y textuales que reúnan la necesidad obvia del Web de extensibilidad.
- El diseño de XML debe ser formal y conciso.
- Los documentos XML deben ser fáciles de crear.
- La simplicidad del marcado XML es de mínima importancia.

XML es un meta-lenguaje, un lenguaje usado para describir otros lenguajes. En otras palabras, XML es una sintaxis usada para describir otros lenguajes de marcado. La Recomendación 1.0 de XML tal como la liberó la W3C, no define un conjunto de etiquetas o palabras clave del lenguaje. El término sintaxis se refiere a un conjunto de restricciones sobre qué y dónde las etiquetas pueden ser puestas, y el rango aceptable de caracteres que son legales, también como las reglas para el marcado en general.

Existen muchos lenguajes de marcado descritos con XML. Por ejemplo, la Recomendación de Firmas XML [7] define un lenguaje de marcado usado para representar una Firma Digital; el Draft de Encriptación XML [9] define un lenguaje de marcado usado para representar elementos encriptados.

Se puede resumir que XML es una plataforma cruzada independiente del software y el hardware que sirve para transmitir información. Es uno de los lenguajes de mayor utilización para realizar transacciones en ambientes Web, y fue una de las bases para el impulso del comercio electrónico.

2.2.1.1 Espacios de Nombres XML (XML Namespaces)

Los Espacios de Nombres XML proporcionan un método para evitar conflictos con nombres de elementos en documentos XML. Como los nombres de los elementos en XML no son fijos, muy frecuentemente pueden ocurrir conflictos con nombres iguales en dos documentos diferentes que describen diferentes tipos de elementos. La forma en que funcionan los Espacios de Nombres XML es la siguiente:

El atributo **namespace** es puesto en la etiqueta inicial de un elemento y tiene la siguiente sintaxis:

```
xmlns:namespace-prefix="namespace"
```

La sintaxis está comprendida por un atributo **xmlns**, por un **prefijo** que estará asociado con el **namespace**.

El Listado 1 proporciona una muestra del uso de los Espacios de Nombres XML. Este caso presenta el Sobre (Envelope) de un Mensaje SOAP reducido, con los elementos necesarios para entender el uso de los Espacios de Nombres XML. El prefijo del Sobre SOAP es `soapenv` y el namespace es `http://schemas.xmlsoap.org/soap/envelope/` definidos en la especificación de SOAP.

Listado 1. Ejemplo de uso de los Espacios de Nombres XML

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
        <soapenv:Body>
                .
                .
                .
          <soapenv:Body>
</soapenv:Envelope>
```

2.2.1.2 Parsers XML

Los parsers XML son procesadores XML, se encargan de leer un documento XML, verificar su sintaxis, reportar cualquier error y permitir un acceso programado al contenido del documento. La Recomendación XML proporciona dos tipos de parsers: los que no realizan validación y los que realizan validación. La Recomendación XML también proporciona dos categorías de documentos XML: los *bien formados* y los *válidos*. Un parser XML debe determinar si el marcado está bien formado y, si un DTD (Document Type Definition) está presente, determinar si el documento XML es válido. El DTD define la gramática y el vocabulario de un lenguaje de marcado, especificando qué es y qué no es permitido que aparezca en un documento. La Recomendación XML no requiere que los documentos XML tengan DTD. Todos los documentos XML deben seguir las reglas para ser bien formados o si no, dichos documentos no son XML.

Un documento XML es considerado *bien formado* si su sintaxis es correcta. Si un documento no está bien formado quiere decir que al verificarlo con un parser, éste reportará errores [1].

Un documento XML es *válido* si es bien formado, tiene un DTD asociado, y cumple con las restricciones expresadas en ese DTD.

La Figura 1 presenta un diagrama de flujo para determinar si un documento está bien formado y si es válido a través de un DTD.

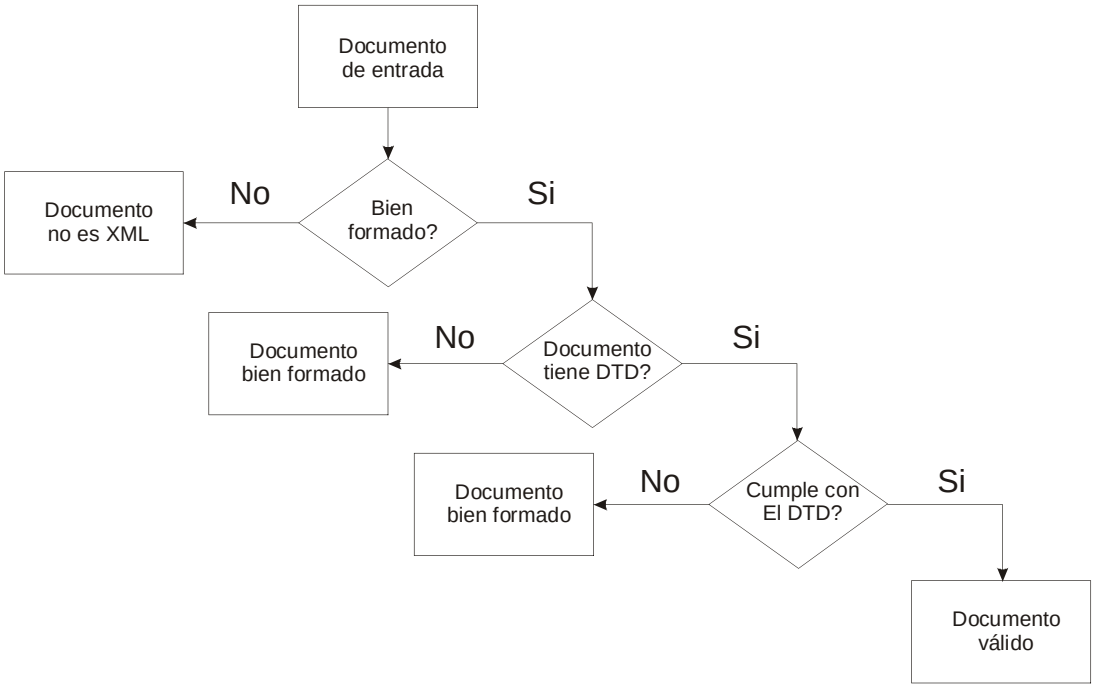

Figura 1. Diagrama de flujo para determinar el tipo de documento XML

2.2.1.3 XML Document Object Model (DOM)

Es una recomendación proporcionada por la W3C para construir estructuras de árbol en memoria a partir de documentos XML [1]. Los parsers que siguen esta recomendación son llamados parsers basados en DOM. La W3C define el DOM como una interfase que permite a los programas y scripts acceder dinámicamente y actualizar el contenido, estructura y estilo de los documentos XML. DOM es importante porque es altamente estandarizado y representa un modelo de programación ampliamente usado y aceptado para documentos estructurados como el XML. Esto hace del API DOM importante porque la mayoría de las implementaciones de Seguridad en XML (Firmas XML o Encriptación XML) tienen soporte de alguna forma u otra para el DOM.

2.2.1.4 Simple API for XML (SAX)

Es una interfase de programación (API, Application Interface Programming) basada en eventos para procesar documentos XML [24].

Originalmente fue una API sólo para Java, fue la primera API ampliamente adoptada para procesar XML en Java, y sigue siendo un estándar “de facto”. Actualmente existen varias versiones para otros lenguajes de programación diferentes de Java.

Los parsers basados en SAX procesan el documento y generan eventos (por ejemplo, notificaciones a la aplicación) cuando etiquetas, texto, comentarios, etc., son encontrados. Estos eventos retornan datos del documento XML. Programas de software pueden “escuchar” los eventos para obtener datos del documento XML.

A diferencia de los parsers XML que usan DOM, SAX no crea nodos individuales de objetos y tampoco construye una jerarquía de árbol DOM en memoria. SAX lee los elementos, anuncia eventos, descarta elementos no requeridos y se mueve al siguiente elemento. Es muy útil cuando se tienen archivos XML grandes y sólo se necesita procesar una porción del archivo, y extraer información de nodos específicos.

2.2.2 WSDL (Web Services Definition Languaje)

Este protocolo se encarga de describir el Web Service cuando es publicado, proporciona una las características sobresalientes de los Web Services: son aplicaciones auto-descriptivas. Para tener esta característica se requiere que el Web Service esté acompañado de información que permita a los desarrolladores emplear el servicio. Estas descripciones son escritas en WSDL, un lenguaje basado en XML a través del cual un Web Service puede informar a otras aplicaciones los métodos que el servicio proporciona y cómo estos métodos pueden ser accesados [1].

La especificación WSDL emergió cuando Microsoft e IBM decidieron combinar sus tecnologías de descripción en un estándar universal. En marzo de 2001, Microsoft, IBM y Ariba mandaron la especificación WSDL 1.1 a la W3C; la W3C [11] es un organismo encargado de regular los estándares para el Web.

Prácticamente todo Web Service publicado en Internet está acompañado por un documento WSDL asociado, el cual lista las capacidades del servicio, establece su localización en el Web y proporciona instrucciones acerca de su uso. Un documento WSDL define las clases de mensajes que un Web Service puede mandar y recibir, también como la especificación de los datos que una llamada de aplicación debe proporcionar al Web Service para que haga su trabajo. Los documentos WSDL también proporcionan información técnica específica que informa a las aplicaciones cómo conectarse y comunicarse con los Web Services sobre HTTP u otro protocolo de comunicación.

Los documentos WSDL se encuentran en lenguaje XML que es leído por aplicaciones, no por humanos. Aunque la estructura de los documentos WSDL puede parecer compleja, las computadoras que leen el WSDL pueden procesar los documentos y extraer la información que necesitan. Además, la mayoría de las herramientas de desarrollo para Web Services generan documentos WSDL automáticamente.

En el **Apéndice A** se muestra un documento WSDL de ejemplo. En este caso se trata de un servicio de conversión de moneda de dos países. Los parámetros para el método getRate, son el país de donde se va a convertir la moneda y el país al cual se va a convertir la moneda. Se puede observar que contiene información de la localización del servicio, la forma de invocar el servicio y la descripción de lo que hace el servicio, entre otras cosas.

2.2.3 SOAP (Simple Object Access Protocol)

IBM, Lotus Development, Microsoft, DevelopMentor y Userland Software desarrollaron y redactaron la especificación SOAP. Fue conceptualizado en 1998 y publicado como SOAP 0.9 en 1999. Las compañías liberaron varias subsecuentes versiones de SOAP antes de mandar el protocolo a la W3C.

SOAP es un protocolo basado en XML que permite a las aplicaciones comunicarse sobre Internet usando documentos XML llamados *Mensajes SOAP*. SOAP es compatible con cualquier modelo de objetos, porque incluye un conjunto base de capacidades necesarias para entornos de comunicaciones. Por ello, SOAP es independiente tanto de plataformas como de lenguajes de programación. SOAP se puede transportar a través de virtualmente cualquier protocolo de aplicación y de transporte. El protocolo de aplicación más comúnmente usado con SOAP es HTTP sobre el protocolo de transporte TCP/IP. Además, SOAP soporta cualquier método de codificación de datos, permitiendo que las aplicaciones basadas en SOAP manden virtualmente cualquier tipo de información en los Mensajes SOAP [1].

Un Mensaje SOAP tiene un Sobre (Envelope) que describe el contenido, el recipiente deseado y requerimientos de procesamiento de un mensaje. Un sobre consiste de dos distintas partes:

1) El opcional elemento Encabezado (Header). Proporciona instrucciones de procesamiento para aplicaciones que reciben el mensaje. Por ejemplo, puede usarse para incorporar información de ruteo, o para propósitos tales como autenticación, administración de transacciones y pagos.
2) El elemento Cuerpo (Body). Contiene datos para la aplicación y el recipiente deseado del mensaje.

La Figura 2 muestra la estructura básica del Mensaje SOAP.

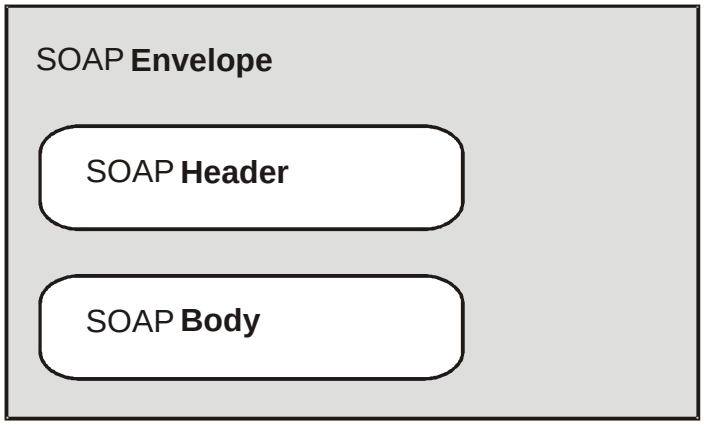

Figura 2. Estructura básica de un Mensaje SOAP

El elemento Cuerpo puede tener opcionalmente un elemento Falla (Fault) que proporciona información sobre los errores que ocurren mientras se procesa el mensaje.

En el Listado 2 se presenta la estructura básica de un Mensaje SOAP. Como reglas de sintaxis un Mensaje SOAP debe tener un Espacio de Nombres para el Sobre y un Espacio de Nombres para el Estilo de Codificación. Así también como se había mencionado anteriormente también debe contener el Encabezado, el Cuerpo y opcionalmente un elemento Falla.

Listado 2. Esqueleto de un Mensaje SOAP

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Header>
  ...
  ...
</soap:Header>
<soap:Body>
  ...
  ...
  <soap:Fault>
    ...
    ...
  </soap:Fault>
</soap:Body>
</soap:Envelope>
```

SOAP impone restricciones sustanciales al XML que usa. En particular, los Mensajes SOAP deben obedecer las siguientes restricciones:

- Los elementos de información con instrucciones de procesamiento están prohibidos (Los elementos de información de un documento XML están definidos en [26]).
- Todos los elementos y atributos deber estar apropiadamente calificados con Espacios de Nombres.
- Los DTDs están prohibidos.
- No se requieren esquemas de procesamiento.

2.2.4 UDDI (Universal Description Discovery and Integration)

IBM, Microsoft y Ariba desarrollaron la especificación UDDI, que define registros en los cuales los negocios pueden publicar información de ellos mismos y los servicios que proveen. Los consumidores pueden usar los registros UDDI para localizar información general y técnica sobre varios proveedores de servicios. Con esta información, los consumidores pueden iniciar transacciones de negocios, formar sociedades y pagar servicios. Los directorios UDDI actúan como una guía telefónica de Web Services [1].

2.3 Integración de las Tecnologías para Web Services

La forma como interactúan cada una de estas tecnologías para proporcionar el servicio se muestra en la Figura 3 [1].

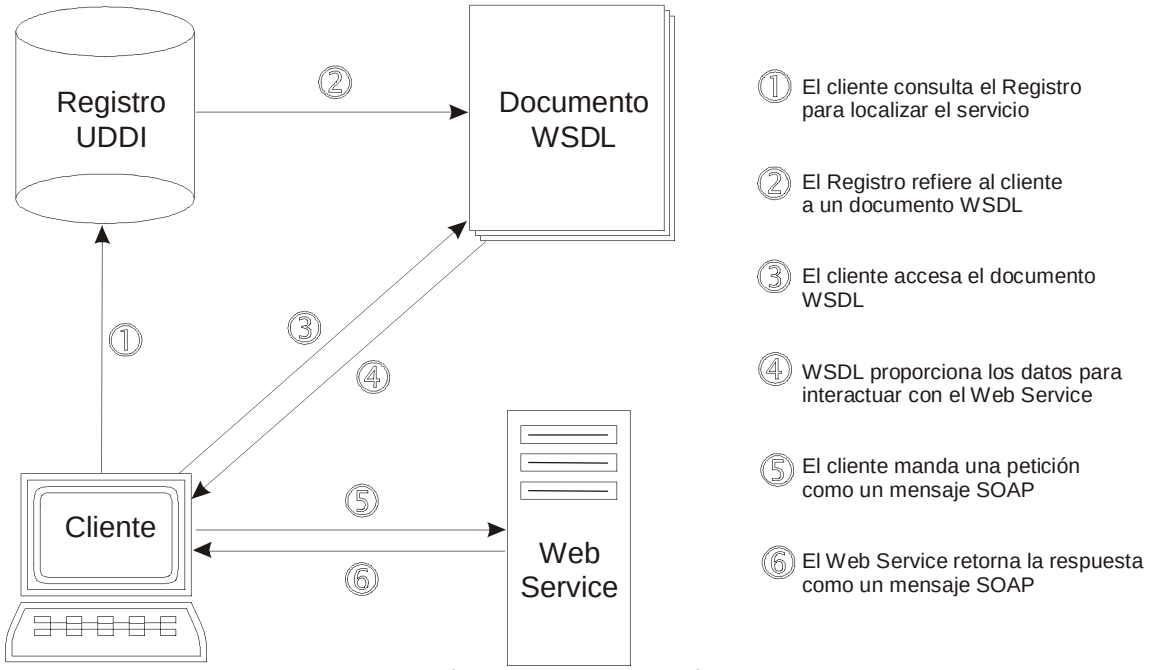

Figura 3. Interacción entre las tecnologías para Web Services

Primeramente el cliente accede al registro UDDI para localizar un Web Service, después de ello el cliente accede al documento WSDL del Web Service, este documento es encontrado por medio del registro UDDI. La información WSDL incluye la ubicación del Web Service y los parámetros que el Web Service necesita para el programa de petición del serivcio. Con la información necesaria para interactuar directamente con el Web Service, el cliente manda una petición como un Mensaje SOAP; el Web Service devuelve la respuesta al cliente como un nuevo Mensaje SOAP. La respuesta retornada por el Web Service no necesariamente tiene que ser inmediata.

2.4 Desempeño de Web Services

Según The British Computer Society, el desempeño puede tener dos significados [5]:

1) La rapidez con la cual una computadora opera, ya sea teóricamente (por ejemplo, usando una fórmula para calcular MTOPS – Millones de instrucciones teóricas por segundo) o contando las operaciones o instrucciones ejecutadas (por ejemplo, MIPS – Millones de instrucciones por segundo) durante una prueba benchmark. El benchmark usualmente involucra trabajo que intenta imitar las clases de trabajo que la computadora hace durante su uso habitual.
2) La efectividad total de un sistema computacional, incluyendo el throughput, tiempo de respuesta individual, y disponibilidad.

Según Harvey Deitel: “el desempeño de los Web Services está comprendido de dos factores principales: el throughput y la latencia. El throughput representa el número de peticiones que un Web Service procesa en un período de tiempo dado, y la latencia representa la longitud de tiempo que el servicio toma para responder cada petición. Un mejor desempeño está indicado con un valor alto de throughput y valor bajo de latencia” [1].

2.4.1 Trabajos relacionados

Existe un principio administrativo universal que expresa: "Lo que no es medible no es administrable, ni controlable, ni auditable", y también se afirma que: “Las mediciones de productividad son un punto de inicio para lograr y mantener un desempeño de clase mundial” [42].

En el ámbito de sistemas computacionales, las compañías que mantienen aplicaciones de negocios, al tener presente esta filosofía, requieren de herramientas de software que permitan medir sus aplicaciones. De esta forma pueden llevar a cabo las actividades de administración, control y auditoría de sus sistemas y plataformas.

Uno de los principales factores que se requieren medir dentro de los sistemas es el desempeño. Como se había mencionado anteriormente, el desempeño de las aplicaciones distribuidas representa la efectividad total del sistema computacional, incluyendo el tiempo de transmisión en la red.

Debido a las necesidades de las corporaciones de llevar a cabo mediciones sobre el desempeño de sus sistemas, han salido a la luz pública distintas herramientas y metodologías para cubrir estas necesidades. A continuación se presentan algunos trabajos relacionados específicamente con el desempeño de Web Services o con alguna de las tecnologías relacionadas con dichas aplicaciones.

1. Mundy y Chadwick realizaron una serie de pruebas para comparar los mensajes XML con mensajes equivalentes escritos en Abstract Syntax Notation One con Basic Encoding Rules (ASN.1/BER) [12]. ASN.1 es un protocolo para la especificación y codificación de estructuras de datos en una representación binaria que se emplea principalmente para enfatizar la eficiencia. En cambio XML es utilizado cuando se requiere un desarrollo fácil de aplicaciones. Las pruebas de este trabajo cubrieron el desempeño de la creación, transmisión, y recuperación en los dos lenguajes mencionados. Para el análisis del desempeño se usaron programas cliente y servidor en Java, los cuales intercambian mensajes usando atributos de certificados. La tarea del cliente es crear el atributo de certificado y transmitirlo al programa servidor a través de sockets. El servidor verifica el certificado y entonces lo parsea en una estructura de datos para tener un acceso fácil a cualquiera de sus elementos. Entre las comparaciones hechas se encuentran el tamaño del bloque de datos, el tiempo de transmisión y los tiempos de codificación y decodificación de los datos.

2. Trade3 es la tercera generación de benchmarks para WebSphere que tiene como objetivo probar el desempeño de las aplicaciones [13]. El Trade3 modela una aplicación en línea para corredores de bolsa que ha sido rediseñada y desarrollada para cubrir las nuevas implementaciones en WebSphere. Proporciona una carga de trabajo de

tiempo real para estas nuevas implementaciones entre las cuales se incluyen los Web Services con sus tecnologías asociadas tales como SOAP, WSDL y UDDI, con el fin de obtener mediciones del desempeño de los componentes en WebSphere.

3. Por la necesidad de contar con un motor de Web Services escalable, en la Universidad de Duke, Vinay Bansal y Piyush Shivam [14] diseñaron una Arquitectura Escalable para Web Services (SWSA, Scalable Web Services Architecture). Según estos autores, la mayoría de los Web Services hoy en día, especialmente los escritos en Java, usan una arquitectura basada en pilas de hilos para tener escalabilidad. Esta arquitectura no se desempeña bien para grandes cantidades de usuarios. Se realizó una evaluación del desempeño para poder comparar los Web Services hechos con estas dos arquitecturas. Los motores de Web Services empleados fueron el Apache Axis y el motor de SWSA. Se realizaron mediciones del tiempo de respuesta y el throughput; para ello se emplearon tres aplicaciones: Un "Web Service Calculadora", un "Web Service Valor de Pi", y un "Web Service de traducción". Los resultados pueden verse en [14].

4. La empresa de software Microsoft realizó una comparación del desempeño de Web Services hechos con sus plataformas ASP.NET y .NET Remoting. La finalidad de la comparación era proporcionar un estudio que permitiera elegir entre estas plataformas según las características de desarrollo y el tipo de servicios que se necesitaran proveer. El throughput y la latencia se emplearon como los indicadores de desempeño. Se empleó para la comparación una máquina cliente y tres servidores, uno funcionaba como servidor web, otro como servidor de aplicaciones y el otro como servidor de bases de datos; tanto el cliente como los servidores son sistemas multiprocesadores. Del lado del cliente se empleó el *Microsoft Application Center Test*, el cual está diseñado para estresar al servidor web y analizar los problemas de desempeño y escalabilidad con las aplicaciones Web. Los resultados pueden verse en [15].

5. Latency Performance of SOAP Implementations. Davis y Parashar realizaron una evaluación experimental de la latencia de varias implementaciones de SOAP operando sobre HTTP, y compara los resultados con el desempeño de JavaRMI, CORBA, HTTP, y con el tiempo de retardo de TCP (setup time). El objetivo de este trabajo es identificar las fuentes de ineficiencia en las implementaciones actuales de SOAP y discutir los cambios que pueden mejorar su desempeño. Las implementaciones SOAP estudiadas incluyen Microsoft SOAP Toolkit, SOAP::Lite Perl module, y Apache SOAP [17].

Los experimentos que se realizaron son los siguientes:

Se probó la llamada a un método, que no contiene instrucciones, sólo para determinar el overhead asociado con una llamada SOAP.
Se probó la llamada a un método que regresa un String. Las pruebas se hicieron con diferentes tamaños de Strings.
Se probó la llamada a un método que regresa un arreglo de enteros. Las pruebas se hicieron con diferentes tamaños de arreglos.

Cada una de las pruebas fue ejecutada con el cliente y el servidor en el mismo host, y luego con el cliente y el servidor en diferentes hosts pero unidos por una pequeña LAN de 10 Mbps con sólo estos dos hosts.

Desde que cada tiempo medido representa 200 llamadas y la precisión del reloj es de 10 microsegundos, la precisión de las mediciones es de +-0.05 milisegundos.

Algunos de los resultados que se presentan, se muestran en la Tabla 1. En este caso se presenta la latencia que genera el envío de un mensaje proveniente de un String entre un cliente y un servidor situados en una misma red.

Tabla 1. GetString() con cliente y servidor separados

System	char[200] Latency (ms)	char[400] Latency (ms)	char[800] Latency (ms)
JavaRMI	1.2	1.5	1.9
CORBA	1.2	1.6	2.0
Apache SOAP 2.2	199.8	199.9	200.0
SOAP::Lite	200.3	200.1	200.1
MS SOAP Toolkit	200.2	200.3	200.3
SoapRMI/Java 1.1	20.3	20.4	20.9

6. A Stream-based Implementation of XML Encryption. En el trabajo [18] se realiza una comparación de la especificación para Cifrado XML de la W3C [9] implementado con parsers DOM y XNI. Usando el API de XNI, se realizó un prototipo de una implementación basada en el flujo de la especificación de la W3C antes mencionada.

Se evaluó el desempeño de cada una de estas implementaciones. Se menciona que el prototipo registra una reducción del 0.27% al 26% en el tiempo de procesamiento para el cifrado de documentos XML con tamaños más grandes que 2 KB, y del 34% al 88% para el descifrado de documentos de cualquier tamaño.

Es importante recalcar que en la mayoría de los trabajos mencionados anteriormente (del 2 al 5) la medición del desempeño no incluye mecanismos de seguridad por lo que toda la información viaja en texto plano. En el Trabajo 1 se emplean Firmas XML con XML Security Suite de IBM [41] para firmar los mensajes XML y una PKI (ver **sección 2.5.6**) de la compañía Entrust (www.entrust.com) para firmar los mensajes ASN.1. En el Trabajo 6 se emplea el Cifrado XML de la W3C.

2.5 Seguridad computacional

Existen distintos aspectos en la seguridad computacional, en algunos sistemas o entornos de aplicación, un aspecto de seguridad puede ser más importante que los otros. El tipo de seguridad que una organización requiere influye en la selección de las técnicas particulares de seguridad y los productos que se necesitan para reunir estos requerimientos [19].

Enfocándonos en la seguridad para aplicaciones distribuidas, existen cinco requerimientos de seguridad para transmisión de mensajes [20]:

- Confidencialidad. Garantiza que ninguna persona ajena a la comunicación pueda tener acceso a la información enviada o recibida.
- Autorización. Garantiza que el emisor está autorizado para mandar un mensaje.
- Integridad. Garantiza que el mensaje no fue modificado accidentalmente o deliberadamente en tránsito.
- Autenticación. Garantiza que el mensaje fue transmitido por un emisor propiamente identificado y no es una réplica de un mensaje previamente transmitido.
- No-repudio. Garantiza que el emisor del mensaje no pueda negar que él mandó dicho mensaje.

Los últimos tres requerimientos están fuertemente relacionados. En particular, el no-repudio implica la autenticación del origen del mensaje, lo cual implica también la integridad de los datos.

2.5.1 Criptografía y Encriptación

La ciencia de la criptografía se refiere al estudio de los métodos para mandar mensajes en secreto de tal forma que solo el recipiente deseado pueda remover el disfraz y leer el mensaje. El mensaje original es llamado *texto plano*, y el mensaje disfrazado es llamado *texto cifrado*. El mensaje final, encapsulado y mandado, es llamado *criptograma*. El proceso de transformar texto plano en texto cifrado es llamado *encriptación* o *cifrado*. El proceso inverso de transformar texto cifrado en texto plano, el cual es realizado por el recipiente quien tiene el conocimiento para remover el disfraz, es llamado *desencriptación* o *descifrado* [27].

Esta sección pretende introducir los conceptos básicos de la criptografía digital moderna, describe la funcionalidad de cada uno de ellos y cómo estos conceptos pueden adaptarse para producir controles sólidos de seguridad.

2.5.2 Encriptación Simétrica

También conocida como criptografía de llave secreta, la encriptación simétrica usa la misma llave secreta para encriptar y desencriptar un mensaje [27] (Figura 4). En este caso, el emisor encripta un mensaje usando la llave secreta, después de ello manda el mensaje encriptado al recipiente deseado, quien desencripta el mensaje usando la misma llave secreta.

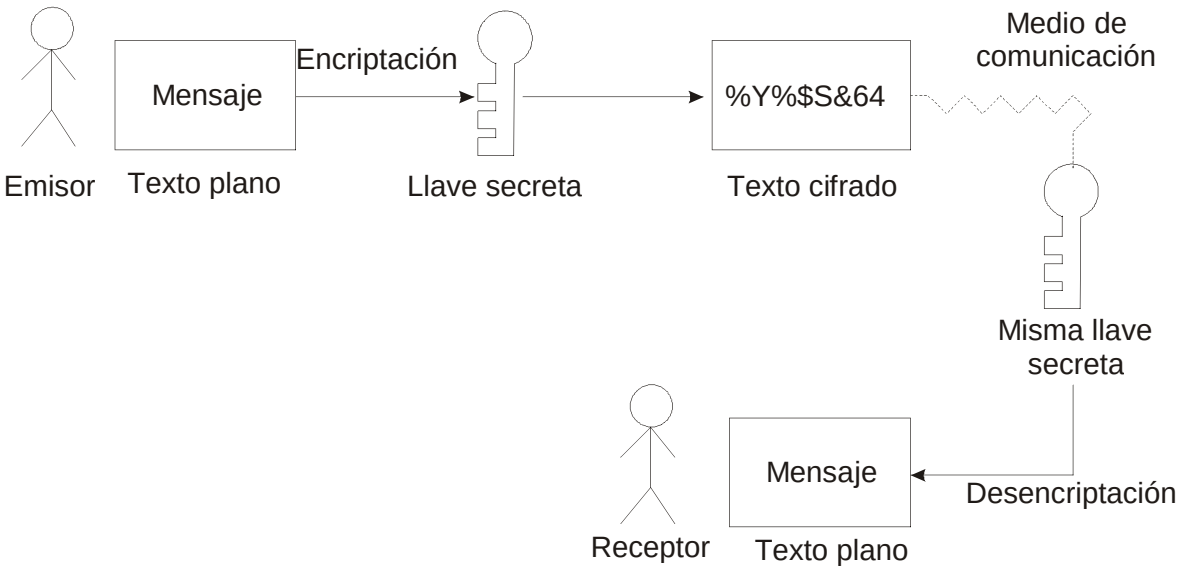

Figura 4. Encriptación y desencriptación usando llave secreta

Un problema fundamental con la criptografía de llave secreta consiste en que antes de que dos personas puedan comunicarse de forma segura, ellas deben encontrar una forma segura de intercambiar la llave secreta. Otro problema es la administración de las llaves, para cada par emisor-receptor debe tenerse una llave diferente. Como resultado, una organización debe tener muchas llaves secretas y administrarlas correctamente.

2.5.3 Encriptación Asimétrica

En 1976, Whitfield Diffie y Martin Hellman, investigadores de la Universidad de Stanford, desarrollaron la criptografía de llave pública para resolver el problema del intercambio seguro de llaves [29]. La criptografía de llave pública también se conoce como encriptación asimétrica. Usa dos llaves inversamente relacionadas: una llave pública y una llave privada. La llave privada es mantenida en secreto por su dueño, mientras que la llave pública es distribuida libremente [27]. Si la llave pública es usada para encriptar un mensaje, sólo la correspondiente llave privada puede desencriptar dicho mensaje (Figura 5). Asumiendo que la llave privada ha sido mantenida en secreto, el mensaje no puede ser leído por nadie más que el receptor deseado. Por ello el sistema asegura la privacidad del mensaje. Aunque las dos llaves son matemáticamente

relacionadas, derivar una de la otra toma enormes cantidades de poder computacional y tiempo, suficientes para desanimar intentos de deducir la llave privada.

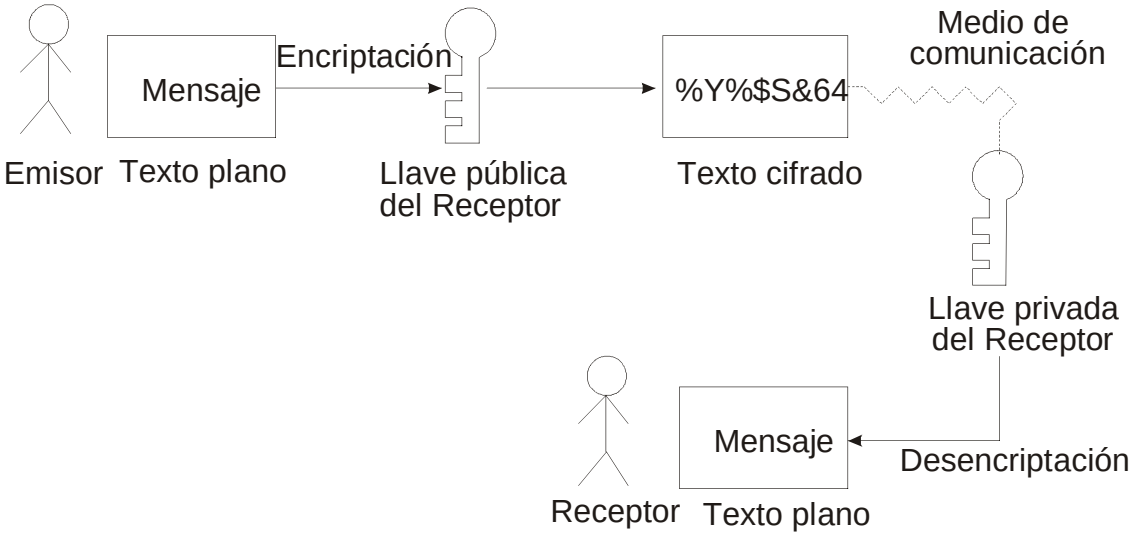

Figura 5. Encriptación y desencriptación usando criptografía de llave pública

2.5.4 Funciones Hash

Una función hash permite obtener un resumen (digest) a partir de un mensaje. Dicho resumen es mucho más pequeño que el mensaje original, y es muy difícil encontrar otro mensaje que tenga el mismo resumen [27].

La función hash presenta las siguientes características:

- Es irreversible, es decir que a partir del resultado de la función hash nunca se podrá obtener el documento original.
- Un ligero cambio en el documento original, genera un resultado de la función hash completamente distinto del documento original.

En el caso común donde un método de autenticación toma gran cantidad de esfuerzo computacional y este esfuerzo es proporcional al número de bits que serán autenticados, se puede asegurar un documento grande autenticando su resumen.

Algunos de los algoritmos más utilizados para generar resúmenes (message digests) se presentan en la Tabla2.

Tabla 2. Algoritmos Hash más populares actualmente

Algoritmo de Hash	Autor(es)	Longitud del resumen
RIPEMD-160 (RACE Integrity Primitives Evaluation Message Digest) [30]	Hans Dobbertin, Antoon Bosselaers, Bart Preneel	160 bits
SHA-1 (Secure Hash Algorithm) [31]	Federal Information Processing Standards	160 bits
MD5 (Message-Digest v.5) [32]	Ronald Rivest	128 bits

2.5.5 Firmas Digitales

Diffie y Hellman en su clásico artículo New Directions in Cryptography [29] fueron los primeros que discutieron el tema de las Firmas Digitales. Manifestaban: "…para desarrollar un sistema capaz de remplazar el actual contrato escrito con una forma de comunicación puramente electrónica, hemos descubierto un fenómeno digital con las mismas propiedades que una firma escrita. Debe ser fácil para cualquiera reconocer la autenticidad de la firma, pero imposible para quien no sea el firmante legítimo producirla."

En el ámbito de Internet, una Firma Digital es un método criptográfico de comunicación que autentica transacciones que se llevan a cabo sobre la Red.

Las Firmas Digitales son el equivalente a la autenticación de mensajes con llave pública [28]. La aplicación que genera la Firma Digital debe tener acceso a los datos que serán firmados y la llave para la generación de la firma. Después, una aplicación puede verificar la firma si tiene acceso a los datos firmados y a la llave para la verificación de la firma (la llave de verificación puede o no ser la misma que la llave de generación, dependiendo de la clase de autenticación en uso). Si la firma es verificada, el verificador sabe si alguna aplicación con la llave de generación ha producido la firma o un adversario ha roto los algoritmos criptográficos. Si asumimos que tenemos buenos algoritmos criptográficos y un buen control sobre quién tiene una copia de la llave, y entonces los datos son firmados en algún lugar y tiempo y después verificados en un diferente lugar o tiempo, se puede confiar en dos cosas:

1. Los datos fueron firmados con la llave.
2. Los datos no han sido modificados de su valor original.

Un ejemplo de configuración para Firmas Digitales se presenta en la Figura 6. En este caso es Alicia quien usa un algoritmo de generación de llaves para generar un par (S_{Alicia}, P_{Alicia}), donde S_{Alicia} es la llave secreta de Alicia y P_{Alicia} es la llave pública de Alicia, esta última llave se publica. Cuando ella quiere mandar un mensaje firmado *m* a Bob, ella computa una firma $s := \sigma(S_{Alicia}, m)$. Ella manda *m* y *s* a Bob. Bob usa un algoritmo de verificación $v(P_{Alicia}, m, s)$ que usa la llave pública de Alicia para verificar la firma.

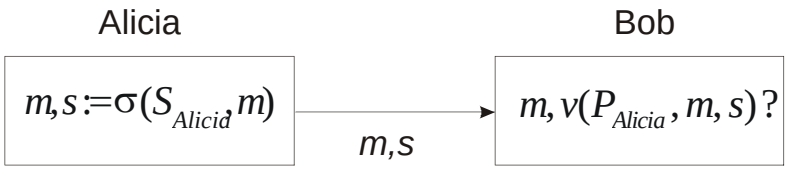

Figura 6. Una configuración para Firmas Digitales

En base a lo descrito anteriormente se puede afirmar que las Firmas Digitales cubren los requerimientos de autenticación, integridad y no repudio necesarios para la transmisión de mensajes.

2.5.6 Infraestructura de Llave Pública, Certificados y Autoridades Certificadoras

Uno de los problemas con la criptografía de llave pública es la administración de las llaves, alguien puede asumir la identidad de otra persona obteniendo su llave y mandando mensajes encriptados con dicha llave.

La Infraestructura de Llave Pública (Public Key Infraestructura, PKI) proporciona una solución a este problema. PKI integra criptografía de llave pública con certificados digitales y autoridades certificadoras para autenticar las partes en una transacción [27].

Los certificados digitales ofrecen una forma de proveer garantías sobre una llave pública y son emitidos por una Autoridad Certificadora (Certificate Authority, CA). En general, los certificados consisten de los siguientes componentes:

- La llave pública.
- Alguna información asociada tal como una identidad o autorización de acceso.
- Una fecha de emisión y expiración.
- Una autenticación de la Firma Digital por una Autoridad Certificadora sobre esta información.

Cualquiera que conociendo y confiando en la llave pública de la Autoridad Certificadora y teniendo el certificado, puede tener confianza en que la llave pública dentro del certificado está asociada con la identidad o debe tener el acceso indicado.

2.5.7 Certificados X.509

La primera versión apareció en 1988 y fue publicada como el formato X.509v1, siendo la propuesta más antigua para una infraestructura de clave pública (PKI) a nivel mundial. Esto junto con su origen ISO/ITU han hecho de X.509 el PKI más ampliamente utilizado. Más tarde fue ampliada en 1993 por la versión 2 únicamente en dos campos, identificando de forma única el emisor y usuario del certificado. La versión 3 de X.509 amplía la funcionalidad del estándar X.509 [33].

2.5.8 Certificados en Windows

En Windows se emplea la versión 3 de la recomendación X.509 de la ITU-T para la sintaxis y el formato de certificados.

Hay cuatro orígenes básicos para los certificados que se encuentran en los almacenes de certificados de un equipo con Windows:

- El certificado se incluye con la instalación de Windows XP y viene en el CD de Windows XP.
- Utiliza una aplicación como Internet Explorer para entrar en una sesión SSL, durante la cual se almacenan certificados en su equipo después de establecer la confianza.
- Decide aceptar explícitamente un certificado, como cuando instala software o recibe correo electrónico cifrado o firmado digitalmente de otras personas.
- Solicita un certificado a una entidad emisora de certificados, como un certificado necesario para tener acceso a recursos organizativos específicos.

En la Figura 7 se muestra desde la Consola de Administración de Microsoft Windows un certificado instalado en el almacén Personal de certificados, este certificado fue emitido para la autenticación del cliente.

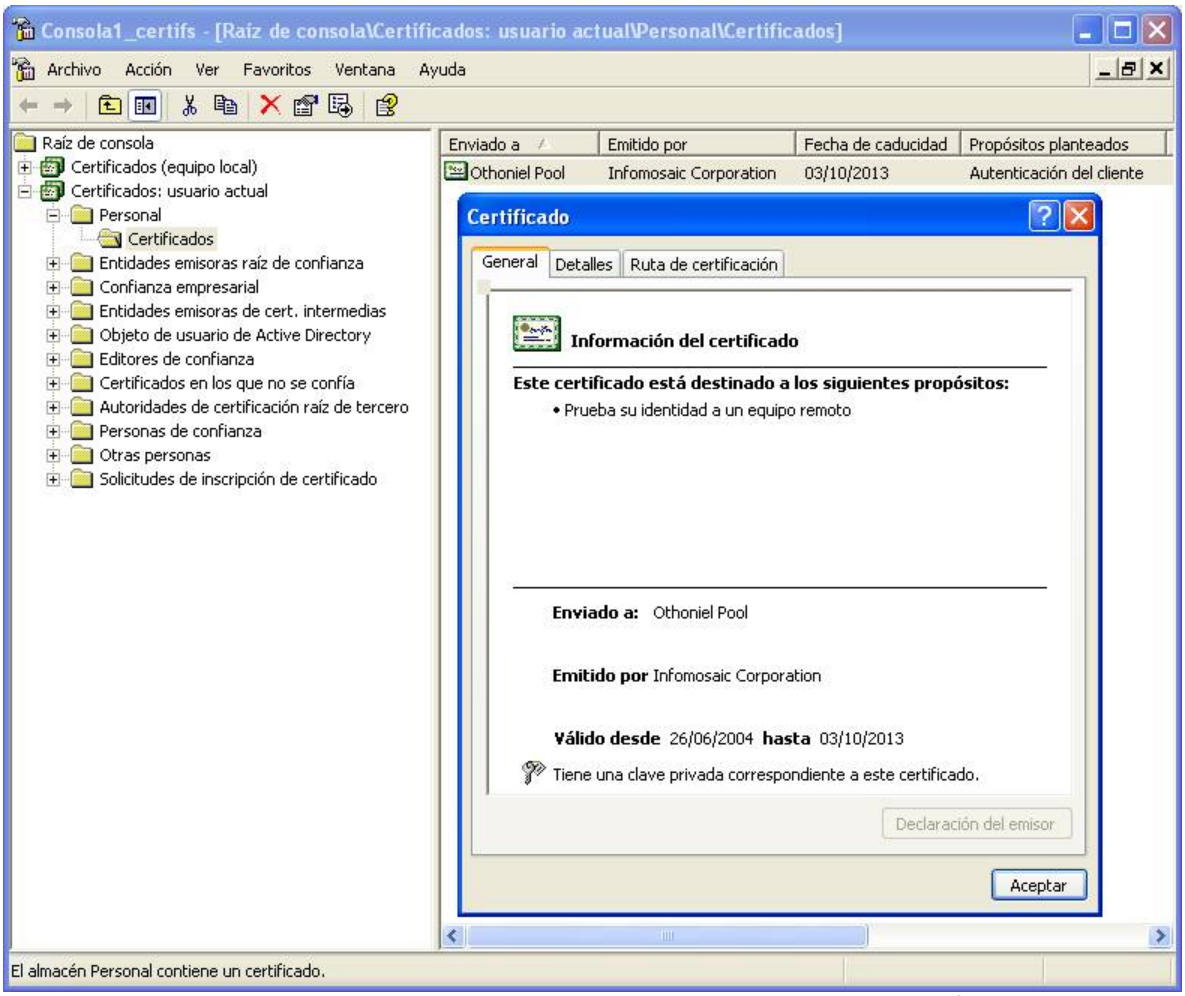

Figura 7. Vista de un certificado Windows desde la Consola de Administración de Windows

2.6 Seguridad para Web Services

Uno de los mayores aspectos dentro de los sistemas distribuidos es el mantener los controles de seguridad necesarios sobre el sistema [39]. En las aplicaciones Web, la importancia de la seguridad se acentúa más. La gran exposición de la Web conlleva a que prácticamente cualquier persona pueda irrumpir en una aplicación Web y poner en riesgo su seguridad.

Los Web Services emplean tecnologías web, por ello muchos de los potenciales ataques contra sitios web son los mismos para Web Services. Uno de los principales problemas con los Web Services es que los mensajes XML contienen, en forma de texto plano, un conjunto concentrado de datos que pueden ser muy interesantes para un adversario. El hecho de que los Mensajes SOAP concentren gran parte de los datos de las transacciones hace que la seguridad en la transmisión sea la más importante de todas [21].

Existen tres áreas de los Web Services donde la seguridad debe ser aplicada. Estas áreas son la autenticación/autorización, seguridad en la capa de transporte, y la seguridad en la capa de aplicación.

Autenticación/Autorización. Es necesario que un Web Service comience un proceso para identificar a un usuario a través de un esquema de autenticación. Estos esquemas determinan qué métodos el usuario tendrá permitido accesar. Existen técnicas que van desde la autenticación con el protocolo HTTP a través de un nombre de usuario y contraseña, hasta métodos más complejos como la autenticación con un sistema Kerberos o biométricas. Kerberos es un protocolo de código abierto que emplea criptografía de llave secreta para autenticar usuarios en una red y mantiene la integridad y privacidad de las comunicaciones en la red. La biometría usa información única personal, tal como huellas digitales, el iris o la cara, para identificar un usuario.

Seguridad en la Capa de Transporte. La seguridad en la capa de transporte es crucial para los Web Services cuando información crítica es transportada a través de redes inseguras como Internet. Determinar qué nivel de seguridad en la capa de transporte será necesario depende de las necesidades básicas del cliente. Entre los protocolos empleados para asegurar las comunicaciones en la capa de transporte se encuentra SSL (Secure Socket Layer), TLS (Transport Layer Security) e IPSec (Internet Protocol Security) en conjunción con las VPN (Virtual Private Networks). SSL es un protocolo no propietario comúnmente usado para asegurar la comunicación entre dos computadoras en Internet y en el Web a través de sockets seguros. El protocolo TLS diseñado por Internet Engineering Task Force, es similar a SSL. Los VPNs sirven para establecer túneles seguros a través de los cuales los datos son pasados entre múltiples redes sobre Internet. IPSec es una de las tecnologías usadas para asegurar el túnel donde pasan los datos, asegurando la privacidad de los datos y la integridad, así como la autenticación de los usuarios.

Seguridad en la Capa de Aplicación. Este tipo de seguridad juega un papel importante en la autenticación y verificación de las partes involucradas en las transacciones XML. Desde que los Web Services son inherentemente transaccionales, se recomienda procurar la identificación de los usuarios, y realizar re-validaciones y re-identificaciones a través del proceso transaccional, de esta forma se puede reforzar la seguridad proporcionada por las otras dos área mencionadas previamente; el empleo de

mecanismos de seguridad en la capa de aplicación es necesario cuando los clientes exigen altos niveles de seguridad a los proveedores de servicios. Existen opciones para usar estándares de criptografía con llave pública que pueden verificar porciones de mensajes contra Certificados Digitales y Firmas Digitales. La Encriptación XML y las Firmas XML en los mensajes son los mecanismos de seguridad con mayor auge en la capa de aplicación.

2.6.1 Firmas Digitales XML

Blake Dournaee define una Firma digital XML como la sintaxis específica usada para representar una Firma Digital sobre cualquier contenido digital. Las Firmas XML dependen de un buen número de tecnologías dispares XML y criptográficas. La sintaxis de Firma Digital XML está diseñada con un alto grado de extensibilidad y flexibilidad; estas nociones se añaden a la naturaleza abstracta de la sintaxis, pero proporcionan una sintaxis de firma que es conducente a casi cualquier operación con firmas [33].

2.6.1.1 La Recomendación W3C para la Sintaxis y el Procesamiento de Firmas XML

La recomendación W3C para la sintaxis y el procesamiento de Firmas XML define la sintaxis para las Firmas XML y sus reglas de procesamiento asociadas. Esta recomendación, como la mayoría de las demás recomendaciones relacionadas con XML, puede ser encontrada en [11]. La recomendación especifica que una Firma XML puede ser aplicada al contenido de uno o más recursos.

Las Firmas XML pueden clasificarse si los datos a ser firmados aparecen en uno de los siguientes lugares:

- Dentro del elemento Firma XML, el cual es llamado *enveloping signature* (Figura 8).
- Fuera del elemento Firma XML, pero circundantes, el cual es llamado *enveloped signature* (Figura 9).
- Fuera y disjuntos del elemento Firma XML, el cual es llamado *detached signature* (Figura 10). Los datos firmados pueden estar en el mismo documento o en otro lugar.

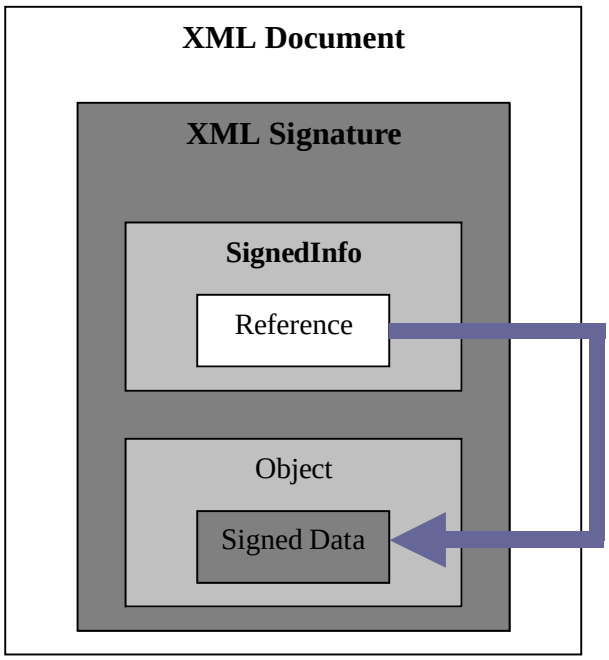

Figura 8. Enveloping signature

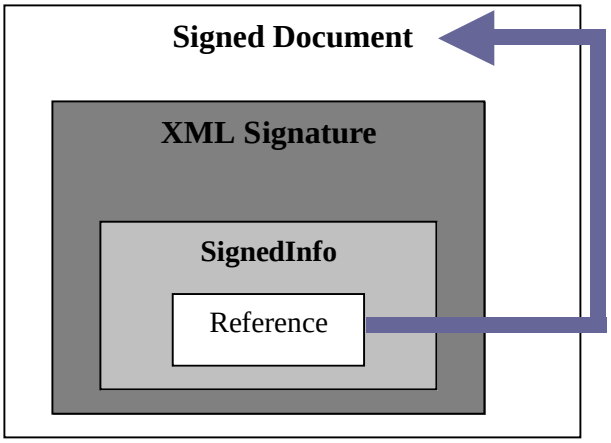

Figura 9. Enveloped signature

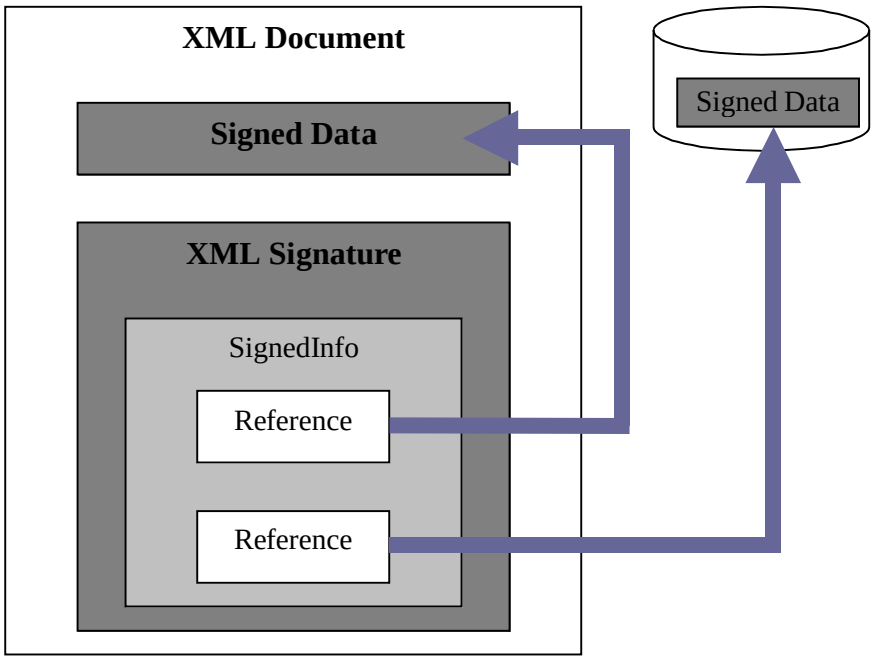

Figura 10. Detached signature de dos fuentes de datos

2.6.1.2 Canonización XML

Es un método para generar una representación física de un documento XML que cuente con cambios permisibles que preserven una equivalencia lógica. Si dos documentos, D y D', tienen la misma forma canónica, entonces los dos documentos son lógicamente equivalentes, bajo un contexto de aplicación específico.

El algoritmo de canonización para documentos XML predeterminado es Canonical XML 1.0 [34], pero cualquier algoritmo aceptable de canonización puede ser usado.

2.6.1.3 El Elemento Signature

Las Firmas Digitales XML se representan por el elemento **Signature**. Una estructura básica del elemento Firma (Signature) puede verse en el Listado 3. Dicho listado proporciona un esqueleto XML de las partes que componen el elemento Firma.

Dentro del elemento Firma, están contenidos la información de la firma (**SignedInfo**), el valor de la firma (**SignatureValue**) y la llave empleada para validar la firma (**KeyInfo**). Dentro de SignedInfo se encuentra el método de canonización (**CanonicalizationMethod**), el método de firmado (**SignatureMethod**) y una Referencia (**Reference**).

La Referencia especifica los datos que se van a firmar a través de su atributo URI (Uniform Resource Identifier). El método de resumen (**DigestMethod**) que indica el algoritmo con el cual se hará el resumen de los datos, y el valor del resumen

(**DigestValue**) están contenidos en el elemento Referencia. Un resumen de datos es análogo a una huella digital única calculada de los datos y que representa a los mismos (ver **sección 2.5.4**). El resumen es más corto que los datos. Para propósitos prácticos, firmar el resumen es tan bueno como firmar los datos originales. El verificador puede verificar la firma sobre el resumen, checando si el resumen corresponde a los datos, y tomar la verificación de la firma como si se hubiera hecho sobre los datos. El elemento Referencia puede contener Transformaciones dentro del elemento opcional **Transforms**. El elemento Transforms proporciona un poderoso mecanismo para indicar el procesamiento de los datos antes de que se realice el resumen. Si se tienen varias transformaciones, la salida de una transformación es la entrada de la siguiente. La salida de la última transformación será la entrada para el proceso de cálculo del resumen. Un ejemplo de transformación sería la extracción de un subconjunto de los datos, si sólo ese subconjunto se desea firmar.

El elemento **KeyInfo** aparece como el hijo de un elemento SignedInfo y proporciona información a un recipiente sobre qué material usar en la validación de una firma o en el descifrado de datos. Este material puede ser llaves, nombres de llaves, información de administración de llaves o certificados digitales.

Listado 3. Estructura XML del Elemento Signature

```
<Signature>
      <SignedInfo>
            (CanonicalizationMethod)
            (SignatureMethod)
            <Reference>
                  (Transforms)
                  (DigestMethod)
                  (DigestValue)
            </Reference>
      </SignedInfo>
      (SignatureValue)
      <KeyInfo>
            <KeyValue>...</KeyValue>
      </KeyInfo>
</Signature>
```

2.6.1.4 Generación de Firmas XML

La recomendación para Firmas XML define dos pasos para la generación de Firmas: la generación de referencias y la generación de la firma. Los pasos para dicha generación son las siguientes:

Generación de Referencias

Para cada recurso que será firmado,

1. Aplicar las transformaciones (si están presentes).
2. Calcular el valor de resumen del recurso transformado.
3. Crear un elemento <Reference>. Esto incluye atributos opcionales, transformaciones, el identificador para el algoritmo de resumen y el valor de resumen actual.

Generación de la Firma

1. Usando las referencias creadas en la generación de referencias, crear un elemento <SignedInfo> que especifique un método de firmado y un método de canonización.
2. Aplicar el método de canonización y el método de firmado al elemento <SignedInfo>, esto producirá <SignatureValue>.
3. Formar el elemento padre <Signature> usando <SignatureValue> y <SignedInfo>, al igual que todos los elementos opcionales y atributos.

2.6.1.5 Validación de Firmas XML

La Validación de Firmas es el proceso inverso al proceso de Generación de Firmas. La Recomendación para Firmas XML define dos pasos para la Validación de Firmas XML: la Validación de Referencias y la Validación de la Firma.

Estas dos fases se ilustran en la Figura 11 por medio de un diagrama de flujo. Los pasos para dicha generación son las siguientes:

Validación de Referencias

Primeramente, el elemento <SignedInfo> debe estar canonizado.

Para cada <Reference> que vaya a ser validado, ejecutar los siguientes pasos:

1. El flujo de datos a ser resumido es obtenido des-referenciando el atributo URI de cada elemento <Reference>. Si el atributo URI no está presente, la aplicación debe conocer la ubicación de la fuente de datos. Los datos finales que serán resumidos son el resultado de transformaciones opcionales que son aplicados en forma de cascada.
2. El flujo de datos obtenido en el paso 1 deben ser resumidos usando la función de hash especificada en el elemento <DigestMethod> para el elemento actual <Reference> a ser procesado.
3. El valor de resumen computado en el paso 2 es comparado contra el contenido del elemento <DigestValue> para el actual elemento <Reference> a ser procesdo. Si estos valores no coinciden, la validación de referencias falló.

Validación de la Firma

1. Recuperar la llave de verificación de un elemento <KeyInfo> o de una fuente conocida por la aplicación.
2. Usando la forma canónica de <SignatureMethod>, determinar el algoritmo de firmado que será usado y calcular un valor de firma sobre la forma canónica del elemento <SignedInfo>. Comparar el valor de firma con el valor dentro del elemento <SignatureValue>. Si estos valores no coinciden, la validación de la firma falló.

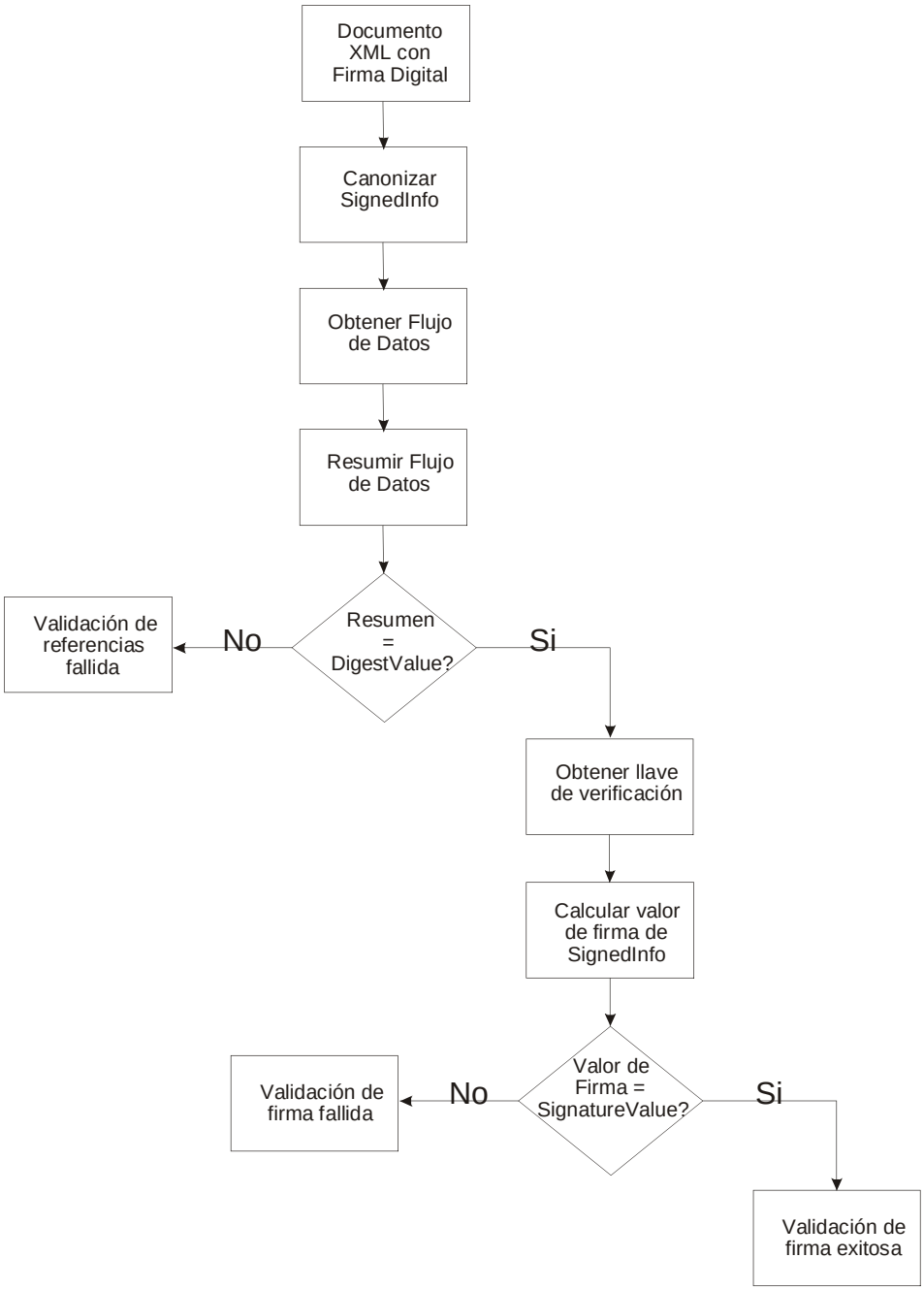

Figura 11. Digrama de Flujo del procedimiento para la Validación de Firmas XML

2.7 Implementaciones SOAP

Varias compañías han producido conjuntos de herramientas para facilitar el rápido desarrollo de Web Services. Estos conjuntos de herramientas permiten a los desarrolladores crear y publicar Web Services sin tener que aprender detalles técnicos de SOAP o WSDL. Los conjuntos de herramientas proporcionan ambientes que ocultan los detalles técnicos de bajo nivel. Esto permite a los programadores desarrollar aplicaciones

rápidamente enfocándose en la funcionalidad de las aplicaciones, en lugar de hacerlo en los detalles de implementación del Web Service. Varios conjuntos de herramientas de desarrollo están disponibles, la mayoría comerciales, pero algunos son de libre distribución.

2.7.1 Apache AXIS

En el 2000, la Fundación de Software Apache (Apache Software Foundation) creó un conjunto de herramientas para SOAP llamada Apache SOAP, el cual estuvo basado en un conjunto de herramientas de IBM, llamado SOAP4J. Apache SOAP implementaba la mayoría de las características incluidas en la especificación de SOAP Versión 1.1. Después de liberar varias versiones de Apache SOAP, Apache creó la herramienta de desarrollo en Java llamada Axis [35].

Al tiempo de esta escritura, la versión más reciente de Apache Axis es la 1.1. Algunas de sus características principales son:

- Axis tiene soporte para SOAP versión 1.1/1.2.
- Permite crear servicios SOAP basados en RPC y mensajes.
- Generación automática de documentos WSDL de servicios implantados.
- Extensiones de seguridad, como encriptación de mensajes y Firmas Digitales.
- Procesamiento de documentos XML a través de SAX, un parser basado en eventos.
- Soporte para varios protocolos tales como HTTP, SMTP, FTP, etc.
- Implantación de programas servidores independientes.
- Implantación de programas servidores adaptables a motores de servlets tales como Tomcat.

Para implementar un Web Service usando este conjunto de herramientas, los desarrolladores primero crean el Web Service en Java. Después de crear el Web Service, Axis proporciona dos opciones para implantar este servicio. Usando la primera opción llamada *Implantación Instantánea* (*Instant Deployment*), los administradores de sistemas pueden publicar un Web Service rápida y fácilmente, pero con limitadas opciones de configuración. Usando la implantación instantánea, Axis expone todos los métodos públicos de la clase Java como Web Services. Sin embargo, los desarrolladores que desean un mejor control pueden usar las características de la *Implantación Personalizada* (*Custom Deployment*), la cual permite a los desarrolladores especificar cuáles métodos exponer. Para personalizar la implantación de un Web Service, los desarrolladores producen un archivo XML especializado, llamado *Descriptor de Implantación de Web Service* (*Web Service Deployment Descriptor, WSDD*). Axis usa el archivo WSDD, para detallar exactamente cuáles métodos exponer.

Axis también ofrece soporte para generar documentos WSDL que pueden ser publicados en un registro. Axis proporciona dos opciones para generar un documento WSDL. En la primera opción, y más simple, Axis genera automáticamente el archivo después de que un Web Service ha sido implantado. La segunda opción permite a los desarrolladores generar documentos WSDL usando una instrucción de línea de comando; esto permite mayor personalización de los archivos WSDL.

2.7.2 webMethods Glue

webMethods Glue es una plataforma fácil de usar, rápida, comprensiva para crear e implantar aplicaciones con Web Services, JSPs (Java Server Pages) y servlets [37]. Glue incluye una implementación compactar y alto desempeño de importantes estándares tales como HTTP, Servlets, XML, SOAP, WSDL y UDDI, e interopera con otras plataformas de Web Services.

Glue puede usarse en dos modos. En *modo independiente*, Glue es esencialmente un servidor de aplicaciones ligero y de alto desempeño. En este modo, Glue usa su propio contenedor de servlets para procesar las peticiones: HTTP, de servlets, y de JSPs. En *modo hospedado*, Glue se adapta a cualquier servidor de aplicaciones o contenedor de servlets de terceros y de esta forma proporciona capacidades avanzadas y de alto desempeño. En este modo, el contenedor de terceras partes toma el control sobre las peticiones: HTTP, de servlets, y de JSPs, dejando a Glue procesar las peticiones SOAP y WSDL. Glue también incluye librerías para clientes Web Services poderosas y de alto desempeño.

Según sus creadores, algunas de sus características principales son:

- Simplicidad. Diseñado para que los desarrolladores creen e implanten Web Services en Java con el menor número posible de líneas de código.
- Apegado a los estándares. Soporta los principales estándares para Web Services y los estándares de seguridad para XML.
- Funcionalidades sofisticadas. Proporciona solución fin-a-fin en el desarrollo de Web Services, con funciones únicas.
- Alta calidad y desempeño. Cuenta con optimizaciones sofisticadas, que hacen de Glue la plataforma más rápida para Web Services.
- Seguridad. Soporta estándares basados en seguridad para encriptación, autenticación de mensajes y WS-Security.

La versión 5.0.1 de Glue tiene soporte para SOAP 1.1 y soporta gran parte de la especificación SOAP 1.2. La versión Profesional de Glue tiene soporte para WS-Security [10], que incluye Firmas Digitales y encriptación.

2.8 Librerías de Firmas XML

Actualmente se encuentran disponibles un buen número de librerías para implementar Firmas XML. En la página Web de Firmas XML de la W3C (http://www.w3.org/Signature/) se encuentra un listado de herramientas y productos para desarrollar aplicaciones basadas en SOAP con Firmas XML. En las siguientes secciones se mencionarán las librerías para Firmas XML: *XML Security* de Apache, *XML Digital Signatures* de Glue y *SecureXML* de Infomosaic, empleadas en el desarrollo del presente trabajo.

2.8.1 XML Security de Apache

El proyecto de Seguridad en XML de Apache Software Foundation tiene el propósito de proporcionar una implementación de los estándares de seguridad para XML. Actualmente el proyecto se enfoca en los estándares de la W3C:

- Sintaxis y Procesamiento de Firmas XML.
- Sintaxis y Procesamiento de Encriptación XML.

El proyecto ha producido dos librerías, una librería en Java y otra en C++. La librería en Java incluye una implementación madura de Firmas Digitales. La Encriptación XML está bajo desarrollo. La librería en C++ funcionalmente es más básica que la librería en Java.

2.8.2 XML Digital Signatures de Glue

Entre los mecanismos de seguridad que proporciona Glue se encuentra el soporte para la especificación WS-Security. Esta especificación es el resultado de un esfuerzo conjunto entre Microsoft, IBM y OASIS para estandarizar la seguridad de SOAP a nivel de mensajes. WS-Security hace uso de la especificación de Firmas XML, la especificación de Encriptación XML y proporciona mecanismos para adjuntar marcas de seguridad (como nombres y passwords) a un mensaje. Hasta esta versión de Glue, datos adjuntos en Mensajes SOAP no pueden ser firmados y en la Encriptación XML sólo algoritmos de llave simétrica pueden ser usados. Los métodos incluidos para la implementación de Firmas XML cumplen con la Recomendación de Firmas Digitales emitida por la W3C.

2.8.3 SecureXML de Infomosaic

Infomosaic SecureXML Digital Signature está diseñado para crear y verificar Firmas Digitales usando la sintaxis XML. El componente implementa la Recomendación W3C para Firmas XML. Negocios confidenciales o información personal, acuerdos o contratos, y transacciones financieras directas, entre otras actividades, necesitan ser aseguradas, firmadas y las identidades participantes verificadas. Infomosaic SecureXML Digital Signature proporciona excelente integridad de los datos, autenticación del mensaje y del firmante y verificación del firmante para todos los tipos de datos, para firmar XML u otros tipos de documentos [38].

SecureXML Digital Signature utiliza certificados PKI X.509v3 los cuales pueden ser almacenados en Windows, Smart Cards o Tokens USB. El componente usa certificados X.509v3 y pares de llaves para construir la Firma Digital. Cuando varios certificados están disponibles se le permite al usuario escoger cual identidad digital usar para la firma.

Capítulo 5

Conclusiones y Trabajo a Futuro

5.1 Conclusiones

Las implantaciones realizadas en el presente trabajo permitieron conocer teórica y prácticamente las tecnologías relacionadas con los Web Services. Asimismo se realizaron distintas implementaciones de la especificación de Firmas XML de la W3C por medio de herramientas disponibles actualmente.

Con el análisis experimental realizado en el presente trabajo se pudieron observar los factores que afectan el desempeño de los Web Services con o sin la implantación de Firmas XML como mecanismo de seguridad. Dicho desempeño comprendido principalmente por el tiempo de respuesta de los servicios implantados.

Los resultados de las pruebas generadas arrojaron que el procesamiento de las Firmas XML es muy costoso en cuanto al tiempo requerido para completar este proceso. Aún así, con un análisis previo de los requerimientos de un sistema puede establecerse el diseño de los servicios de tal forma que el procesamiento de las Firmas XML no sobrepase las exigencias establecidas.

Puede afirmarse que la implementación de Firmas XML es funcional sobre Mensajes con un número bajo de elementos XML. En caso de ser posible para poder emplear la tecnología de Firmas XML será necesario fragmentar los Mensajes con gran cantidad de elementos XML en varios Mensajes más pequeños, de esta forma se puede reducir considerablemente el tiempo de respuesta.

Se pudo demostrar también, que el tamaño de los Mensajes por sí solo no afecta directamente el tiempo requerido para procesar las Firmas XML. El número de elementos XML sí es un factor que modifica severamente el tiempo de procesamiento de las Firmas XML y en consecuencia el tiempo de respuesta de los servicios.

Después de poco más de dos años y medio que fue publicado como Recomendación la Sintaxis y Procesamiento de Firmas XML, aún no ha madurado completamente esta tecnología, pero se espera que en los próximos años existan implementaciones que permitan desarrollar Web Services con Firmas XML eficientes y que proporcionen el requerimiento de escalabilidad.

5.2 Trabajo a futuro

El presente trabajo no estuvo enfocado en los factores que afectan el tiempo de respuesta en el nivel de red. Tampoco tenía como objetivo principal optimizar el desempeño de los Web Services. Para un análisis más exhaustivo se propone estudiar los factores que afectan a nivel de red y a nivel de host el tiempo de respuesta de los Web Services.

Por otro lado, existe un requerimiento importante que las Firmas Digitales por sí mismas no cubren: la confidencialidad. Los Mensajes viajan en texto plano y pueden ser vistos por intrusos. La confidencialidad puede proporcionarse a través de la encriptación de los Mensajes, este requerimiento no fue incluido en el presente trabajo. Se sugiere como trabajo a futuro la implementación de la Recomendación para el Cifrado XML en conjunción con las Firmas XML para su posterior análisis siguiendo la metodología descrita en el presente documento. De esta forma se tendrá mayor seguridad en la transmisión de Mensajes empleados por los Web Services y se podrá observar el desempeño conjunto de estas dos especificaciones.

Apéndice A

Documento WSDL para acceder al servicio getRate

```
<?xml version='1.0' encoding='UTF-8'?>
<!--generated by Glue Professional 5.0.1 on Fri Jul 02 11:46:14 CDT 2004-
->
<wsdl:definitions name='Exchange'
targetNamespace='http://www.webmethods.com/wsdl/Exchange/'
xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
xmlns:xsd='http://www.w3.org/2001/XMLSchema'
xmlns:tns='http://www.webmethods.com/wsdl/Exchange/'
xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
xmlns:tme='http://www.webmethods.com/'>
  <wsdl:message name='getRate0In'>
    <wsdl:part name='country1' type='xsd:string'>
      <wsdl:documentation>The country to convert from
      </wsdl:documentation>
    </wsdl:part>
    <wsdl:part name='country2' type='xsd:string'>
      <wsdl:documentation>The country to convert to</wsdl:documentation>
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name='getRate0Out'>
    <wsdl:part name='Result' type='xsd:float'>
      <wsdl:documentation>The exchange rate </wsdl:documentation>
    </wsdl:part>
  </wsdl:message>
  <wsdl:portType name='Exchange'>
    <wsdl:operation name='getRate' parameterOrder='country1 country2'>
      <wsdl:documentation>Return the exchange rate between two countries
      </wsdl:documentation>
      <wsdl:input name='getRate0In' message='tns:getRate0In'/>
      <wsdl:output name='getRate0Out' message='tns:getRate0Out'/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name='Exchange' type='tns:Exchange'>
    <soap:binding style='rpc'
transport='http://schemas.xmlsoap.org/soap/http'/>
    <tme:optimizations tag='1' href='1' env='1'/>
    <wsdl:operation name='getRate'>
      <soap:operation soapAction='getRate' style='rpc'/>
      <wsdl:input name='getRate0In'>
        <soap:body use='encoded' namespace='x'
encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
      </wsdl:input>
      <wsdl:output name='getRate0Out'>
        <soap:body use='encoded' namespace='x'
encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name='Exchange'>
    <wsdl:port name='Exchange' binding='tns:Exchange'>
```

```
      <soap:address location='http://10.16.86.195:8004/glue/exchange'/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Apéndice B

Mensaje generado con la implementación Axis y la librería XML Security de Apache

```
<soapenv:Envelope soapenv:actor="some-uri" soapenv:mustUnderstand="1"
xmlns:SOAP-SEC="http://schemas.xmlsoap.org/soap/security/2000-12"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <soapenv:Header>
  <SOAP-SEC:Signature>
   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
     <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-
xml-c14n-20010315"/>
     <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
     <ds:Reference URI="#Body">
      <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue>
IVvbdeOXLwmf3cYUcJzwJNHV4Dg=
      </ds:DigestValue>
     </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>
S0MRAxf7LshJuve4ceg8FjQ7h6YnW6e/Ua+T4ExW4RH29j78Gjx2W4M9Ly0tilhEaFwbeM6cm
hG3wh9pfGSlsApLRFz6T3noA8B9SuoXzeXeumuScU/NXdjzR8qplAhGrkeS73oBl7e7oInEZ7
rpNIF3hDcxiTr64QLJAGgYJeU=
    </ds:SignatureValue>
    <ds:KeyInfo>
     <ds:X509Data>
      <ds:X509Certificate>
MIICUjCCAbsCBEDPJq4wDQYJKoZIhvcNAQEEBQAwcDELMAkGA1UEBhMCdXMxDjAMBgNVBAgTB
XRleGFzMQ8wDQYDVQQHEwZkYWxsYXMxEzARBgNVBAoTCndlYm1ldGhvZHMxFDASBgNVBAsTC2
RldmVsb3BtZW50MRUwEwYDVQQDEwxncmFoYW0gZ2xhc3MwHhcNMDQwNjE1MTY0MTE4WhcNMDQ
wOTEzMTY0MTE4WjBwMQswCQYDVQQGEwJ1czEOMAwGA1UECBMFdGV4YXMxDzANBgNVBAcTBmRh
bGxhczETMBEGA1UEChMKd2VibWV0aG9kczEUMBIGA1UECxMLZGV2ZWxvcG1lbnQxFTATBgNVB
AMTDGdyYWhhbSBnbGFzczCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAxkzlZmkDGtecNl
dSRYKYwv7R4O5K0k2gjdfdr+9W5ysHkeiRDyqcfv/x4lNSME8TneBf2pgMyQCS8dzxRL6VeaL
tk7imS/atzqq2Wiw6xjR0eBdjlEbjODZpRccU2iwysa0YkyV+bkb1IfzXLppguhprgo8e/xP6
Pb+t2cmFSeUCAwEAATANBgkqhkiG9w0BAQQFAAOBgQBKlT1TNvNU7zw9aIwDjEiofr4JfLd8j
1FfZUwU+z+jhnvJwy+Ip+plNQUj1K94gtxPWHOb+wSsLMxFsFqN9z/V2ob1XWEYyKVQ2UlMRA
R1USmC1zg6GSZ67/DvWjZmelF3QZsvpRTWccIR5ABzq+8VXgBInzfAVwWNrgD/T16cOA==
      </ds:X509Certificate>
     </ds:X509Data>
     <ds:KeyValue>
      <ds:RSAKeyValue>
       <ds:Modulus>
xkzlZmkDGtecNldSRYKYwv7R4O5K0k2gjdfdr+9W5ysHkeiRDyqcfv/x4lNSME8TneBf2pgMy
QCS8dzxRL6VeaLtk7imS/atzqq2Wiw6xjR0eBdjlEbjODZpRccU2iwysa0YkyV+bkb1IfzXLp
pguhprgo8e/xP6Pb+t2cmFSeU=
```

```
       </ds:Modulus>
       <ds:Exponent>
AQAB
       </ds:Exponent>
      </ds:RSAKeyValue>
     </ds:KeyValue>
    </ds:KeyInfo>

   </ds:Signature>
  </SOAP-SEC:Signature>
  </soapenv:Header>
  <soapenv:Body Id="Body">
    <ns1:getIntegersResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://localhost:8080/MiServicio">
      <ns1:getIntegersReturn soapenc:arrayType="xsd:int[5]"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="soapenc:Array">

        <item>
0    </item>

        <item>
1    </item>

        <item>
2    </item>

        <item>
3    </item>

        <item>
4    </item>

      </ns1:getIntegersReturn>
    </ns1:getIntegersResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Mensaje generado con la implementación Axis y la librería SecureXML de Infomosaic

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <soapenv:Body>
    <ns1:getIntegersResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://localhost:8080/MiServicio">

      <ns1:getIntegersReturn soapenc:arrayType="xsd:int[5]"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="soapenc:Array">

        <item>
0    </item>

        <item>
1    </item>

        <item>
2    </item>

        <item>
3    </item>

        <item>
4    </item>

      </ns1:getIntegersReturn>

    </ns1:getIntegersResponse>

  </soapenv:Body>

 <Signature Id="MySignature" xmlns="http://www.w3.org/2000/09/xmldsig#">
<!--XML Signature produced by Infomosaic SecureXML,
http://www.infomosaic.net-->

  <SignedInfo>
   <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-
c14n-20010315#WithComments"/>
   <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha1"/>
   <Reference URI="">
    <Transforms>
     <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/>
    </Transforms>
    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <DigestValue>
Z6DNQZ/H5A9A4X3rvJHMcOiS4bg=
    </DigestValue>
```

```
   </Reference>
  </SignedInfo>
  <SignatureValue>
P9B8xmYjzRqw0DZ2P0adygfLzeA8zrAlOq1UAJqEui9kS1RC7+EzyY4UkiNvS6tweKfdRBRzO
EEo+CbfW/Y4Pnn0g8dqgRvd4Xi6lj79zuKgM/5/JVgSyI/ZdyuTA0jozAIVZWQRfcsidnH7Ye
shtMjGqKmKqIKMRti/ssOIyEU=
  </SignatureValue>
  <KeyInfo>
   <KeyValue>
    <RSAKeyValue>
     <Modulus>
vIOYtpvSHR1n710urnTH0nj4ttJ2R0cQIt06f5qWAzS7bcuCpOBRk+ngeNoUxr2SGJSmYRajv
bCznOx4+iXSWJFiidbdv8kbAK8VrBqldLiZ2plYBillVBhNcnjoAC9dwyTBJssCah9n21+m3f
iheEYh0mGbD+8OXDWNYmUu280=
     </Modulus>
     <Exponent>
AQAB     </Exponent>
    </RSAKeyValue>
   </KeyValue>
   <X509Data>
    <X509Certificate>
MIIFsTCCBJmgAwIBAgIKWJ7dcAABAAAJ9zANBgkqhkiG9w0BAQUFADCBvjEiMCAGCSqGSIb3D
QEJARYTaW5mb0BpbmZvbW9zYWljLmNvbTELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAkNBMRQwEg
YDVQQHEwtTYW50YSBDbGFyYTEfMB0GA1UEChMWSW5mb21vc2FpYyBDb3Jwb3JhdGlvbjEmMCQ
GA1UECxMdTWFraW5nIERpZ2l0YWwgU2lnbmF0dXJlIEVhc3kxHzAdBgNVBAMTFkluZm9tb3Nh
aWMgQ29ycG9yYXRpb24wHhcNMDQwNjI2MjEzNDA1WhcNMTMxMDAzMjE1OTUzWjCBvDEpMCcGC
SqGSIb3DQEJARYaYWw3ODYwNzBAbWFpbC5tdHkuaXRlc20ubXgxCzAJBgNVBAYTAlVTMQswCQ
YDVQQIEwJDQTEUMBIGA1UEBxMLU2FudGEgQ2xhcmExHzAdBgNVBAoTFkluZm9tb3NhaWMgQ29
ycG9yYXRpb24xJjAkBgNVBAsTHU1ha2luZyBEaWdpdGFsIFNpZ25hdHVyZSBFYXN5MRYwFAYD
VQQDEw1PdGhvbmllbCBQb29sMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC8g5i2m9IdH
WfvXS6udMfSePi20nZHRxAi3Tp/mpYDNLtty4Kk4FGT6eB42hTGvZIYlKZhFqO9sLOc7Hj6Jd
JYkWKJ1t2/yRsArxWsGqV0uJnamVgGKWVUGE1yeOgAL13DJMEmywJqH2fbX6bd+KF4RiHSYZs
P7w5cNY1iZS7bzQIDAQABo4ICMzCCAi8wDgYDVR0PAQH/BAQDAgTwMBMGA1UdJQQMMAoGCCsG
AQUFBwMCMB0GA1UdDgQWBBSRr86A4ivVH8MO7ASBtXxByaMQHTCB+gYDVR0jBIHyMIHvgBTIO
h7z8s+f5sAIUDtq6uWM/k217qGBxKSBwTCBvjEiMCAGCSqGSIb3DQEJARYTaW5mb0BpbmZvbW
9zYWljLmNvbTELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAkNBMRQwEgYDVQQHEwtTYW50YSBDbGF
yYTEfMB0GA1UEChMWSW5mb21vc2FpYyBDb3Jwb3JhdGlvbjEmMCQGA1UECxMdTWFraW5nIERp
Z2l0YWwgU2lnbmF0dXJlIEVhc3kxHzAdBgNVBAMTFkluZm9tb3NhaWMgQ29ycG9yYXRpb26CE
CokOov0wc6NSmZItYNeaGcwXgYDVR0fBFcwVTBToFGgT4ZNaHR0cDovL2luZm9tb3NhaWMtc3
J2ci5pbmZvbW9zYWljLm5ldC9DZXJ0RW5yb2xsL0luZm9tb3NhaWMlMjBDb3Jwb3JhdGlvbi5
jcmwwgYsGCCsGAQUFBwEBBH8wfTB7BggrBgEFBQcwAoZvaHR0cDovL2luZm9tb3NhaWMtc3J2
ci5pbmZvbW9zYWljLm5ldC9DZXJ0RW5yb2xsL2luZm9tb3NhaWMtc3J2ci5pbmZvbW9zYWljL
m5ldF9JbmZvbW9zYWljJTIwQ29ycG9yYXRpb24oMSkuY3J0MA0GCSqGSIb3DQEBBQUAA4IBAQ
A8zm9fe6NT8cxOreo6aPjxV7R9uR3nM0AUAItOr2hv4QNuO1D5kYpZA9vsE/MjMq/NxywZ/v5
a5f7wmCADISs1tzZcDzCvswAvYC6FGJ7OszfTv8rMPCZilhMTBTrbR7R4E3SjFE717R6OcM/P
f2SIwqgNDN9YcClEcq5pP896TP7GeIitSpcMVr7ISrS+dYXEAr0jo3BIwMBswk+qSeA+f8aom
fWw6I1NZqU9JYGfSM+eLV9+7XhSFCC3lGt+x814GzJfl4SkMMhWuh0+59LrVtuZEdKpbWprZR
j8umyvfhOV1cyVODhX/DLmlH97g0nExEuGSG2QMdl4gCLSufia
    </X509Certificate>
   </X509Data>
  </KeyInfo>
 </Signature>
</soapenv:Envelope>
```

Mensaje generado con la implementación Glue y la librería SecureXML de Infomosaic

```
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'>
  <soap:Body
soap:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
    <getIntegersResponse xmlns:n='http://mio'>
      <Result arrayType='xsd:int[5]'>
        <i>0</i>
        <i>1</i>
        <i>2</i>
        <i>3</i>
        <i>4</i>
      </Result>
    </getIntegersResponse>
  </soap:Body>
  <Signature xmlns='http://www.w3.org/2000/09/xmldsig#' Id='MySignature'>
  <!--XML Signature produced by Infomosaic SecureXML,
http://www.infomosaic.net-->
    <SignedInfo>
      <CanonicalizationMethod Algorithm='http://www.w3.org/TR/2001/REC-
xml-c14n-20010315#WithComments'/>
      <SignatureMethod Algorithm='http://www.w3.org/2000/09/xmldsig#rsa-
sha1'/>
      <Reference URI=''>
        <Transforms>
          <Transform
Algorithm='http://www.w3.org/2000/09/xmldsig#enveloped-signature'/>
        </Transforms>
        <DigestMethod
Algorithm='http://www.w3.org/2000/09/xmldsig#sha1'/>
        <DigestValue>kCoSxoLWBfJEEhC/WsQksS7lkic=</DigestValue>
      </Reference>
    </SignedInfo>

<SignatureValue>LBniVAWGEl2Mufce5+0xBnvzhmB2SsClTK8nSbWIWdMBCxaA7Yd2JPGnJ
uJy5Fx2zvvSSfnS4MDqwWvY5c6BmEZZznmKCb5COjiM2ZOuatA9AdZqOfsnNaCSPZWmO3cb4R
08sDva6Ds2+pwC37VknwqyTtUhOoSozEAOPSF7Yho=</SignatureValue>
      <KeyInfo>
        <KeyValue>
          <RSAKeyValue>

<Modulus>vIOYtpvSHR1n710urnTH0nj4ttJ2R0cQIt06f5qWAzS7bcuCpOBRk+ngeNo
Uxr2SGJSmYRajvbCznOx4+iXSWJFiidbdv8kbAK8VrBqldLiZ2plYBillVBhNcnjoAC9dwyTB
JssCah9n21+m3fiheEYh0mGbD+8OXDWNYmUu280=</Modulus>
            <Exponent>AQAB</Exponent>
          </RSAKeyValue>
        </KeyValue>
      <X509Data>
<X509Certificate>MIIFsTCCBJmgAwIBAgIKWJ7dcAABAAAJ9zANBgkqhkiG9w0BAQUFADCB
vjEiMCAGCSqGSIb3DQEJARYTaW5mb0BpbmZvbW9zYWljLmNvbTELMAkGA1UEBhMCVVMxCzAJB
gNVBAgTAkNBMRQwEgYDVQQHEwtTYW50YSBDbGFyYTEfMB0GA1UEChMWSW5mb21vc2FpYyBDb3
Jwb3JhdGlvbjEmMCQGA1UECxMdTWFraW5nIERpZ2l0YWwgU2lnbmF0dXJlIEVhc3kxHzAdBgN
VBAMTFkluZm9tb3NhaWMgQ29ycG9yYXRpb24wHhcNMDQwNjI2MjEzNDA1WhcNMTMxMDAzMjE1
OTUzWjCBvDEpMCcGCSqGSIb3DQEJARYaYWw3ODYwNzBAbWFpbC5tdHkuaXRlc20ubXgxCzAJB
gNVBAYTAlVTMQswCQYDVQQIEwJDQTEUMBIGA1UEBxMLU2FudGEgQ2xhcmExHzAdBgNVBAoTFk
```

```
luZm9tb3NhaWMgQ29ycG9yYXRpb24xJjAkBgNVBAsTHU1ha2luZyBEaWdpdGFsIFNpZ25hdHV
yZSBFYXN5MRYwFAYDVQQDEw1PdGhvbmllbCBQb29sMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCB
iQKBgQC8g5i2m9IdHWfvXS6udMfSePi20nZHRxAi3Tp/mpYDNLtty4Kk4FGT6eB42hTGvZIYl
KZhFqO9sLOc7Hj6JdJYkWKJ1t2/yRsArxWsGqV0uJnamVgGKWVUGE1yeOgAL13DJMEmywJqH2
fbX6bd+KF4RiHSYZsP7w5cNY1iZS7bzQIDAQABo4ICMzCCAi8wDgYDVR0PAQH/BAQDAgTwMBM
GA1UdJQQMMAoGCCsGAQUFBwMCMB0GA1UdDgQWBBSRr86A4ivVH8MO7ASBtXxByaMQHTCB+gYD
VR0jBIHyMIHvgBTIOh7z8s+f5sAIUDtq6uWM/k217qGBxKSBwTCBvjEiMCAGCSqGSIb3DQEJA
RYTaW5mb0BpbmZvbW9zYWljLmNvbTELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAkNBMRQwEgYDVQ
QHEwtTYW50YSBDbGFyYTEfMB0GA1UEChMWSW5mb21vc2FpYyBDb3Jwb3JhdGlvbjEmMCQGA1U
ECxMdTWFraW5nIERpZ2l0YWwgU2lnbmF0dXJlIEVhc3kxHzAdBgNVBAMTFkluZm9tb3NhaWMg
Q29ycG9yYXRpb26CECokOov0wc6NSmZItYNeaGcwXgYDVR0fBFcwVTBToFGgT4ZNaHR0cDovL
2luZm9tb3NhaWMtc3J2ci5pbmZvbW9zYWljLm5ldC9DZXJ0RW5yb2xsL0luZm9tb3NhaWMlMj
BDb3Jwb3JhdGlvbi5jcmwwgYsGCCsGAQUFBwEBBH8wfTB7BggrBgEFBQcwAoZvaHR0cDovL2l
uZm9tb3NhaWMtc3J2ci5pbmZvbW9zYWljLm5ldC9DZXJ0RW5yb2xsL2luZm9tb3NhaWMtc3J2
ci5pbmZvbW9zYWljLm5ldF9JbmZvbW9zYWljJTIwQ29ycG9yYXRpb24oMSkuY3J0MA0GCSqGS
Ib3DQEBBQUAA4IBAQA8zm9fe6NT8cxOreo6aPjxV7R9uR3nM0AUAItOr2hv4QNuO1D5kYpZA9
vsE/MjMq/NxywZ/v5a5f7wmCADISs1tzZcDzCvswAvYC6FGJ7OszfTv8rMPCZilhMTBTrbR7R
4E3SjFE717R6OcM/Pf2SIwqgNDN9YcClEcq5pP896TP7GeIitSpcMVr7ISrS+dYXEAr0jo3BI
wMBswk+qSeA+f8aomfWw6I1NZqU9JYGfSM+eLV9+7XhSFCC3lGt+x814GzJfl4SkMMhWuh0+5
9LrVtuZEdKpbWprZRj8umyvfhOV1cyVODhX/DLmlH97g0nExEuGSG2QMdl4gCLSufia</X509
Certificate>
      </X509Data>
    </KeyInfo>
  </Signature>
</soap:Envelope>
```

Mensaje generado con la implementación Glue y la librería de Firmas XML de Glue

```
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'
xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/'
xmlns:xsd='http://www.w3.org/2001/XMLSchema'>
  <soap:Header>
    <wsse:Security
xmlns:wsse='http://schemas.xmlsoap.org/ws/2002/12/secext'>
      <wsse:BinarySecurityToken ValueType='wsse:X509v3'
EncodingType='wsse:Base64Binary'
xmlns:wsu='http://schemas.xmlsoap.org/ws/2002/07/utility'
wsu:Id='electric-id-B428A2B2-DA90-C072-F2BF-F2DBF412D5FF'>
MIICUjCCAbsCBEDPJq4wDQYJKoZIhvcNA
QEEBQAwcDELMAkGA1UEBhMCdXMxDjAMBgNVBAgTBXRleGFzMQ8wDQYDVQQHEwZkYWxsYXMxEz
ARBgNVBAoTCndlYm1ldGhvZHMxFDASBgNVBAsTC2RldmVsb3BtZW50MRUwEwYDVQQDEwxncmF
oYW0gZ2xhc3MwHhcNMDQwNjE1MTY0MTE4WhcNMDQwOTEzMTY0MTE4WjBwMQswCQYDVQQGEwJ1
czEOMAwGA1UECBMFdGV4YXMxDzANBgNVBAcTBmRhbGxhczETMBEGA1UEChMKd2VibWV0aG9kc
zEUMBIGA1UECxMLZGV2ZWxvcG1lbnQxFTATBgNVBAMTDGdyYWhhbSBnbGFzczCBnzANBgkqhk
iG9w0BAQEFAAOBjQAwgYkCgYEAxkzlZmkDGtecNldSRYKYwv7R4O5K0k2gjdfdr+9W5ysHkei
RDyqcfv/x4lNSME8TneBf2pgMyQCS8dzxRL6VeaLtk7imS/atzqq2Wiw6xjR0eBdjlEbjODZp
RccU2iwysa0YkyV+bkb1IfzXLppguhprgo8e/xP6Pb+t2cmFSeUCAwEAATANBgkqhkiG9w0BA
QQFAAOBgQBKlT1TNvNU7zw9aIwDjEiofr4JfLd8j1FfZUwU+z+jhnvJwy+Ip+plNQUj1K94gt
xPWHOb+wSsLMxFsFqN9z/V2ob1XWEYyKVQ2UlMRAR1USmC1zg6GSZ67/DvWjZmelF3QZsvpRT
WccIR5ABzq+8VXgBInzfAVwWNrgD/T16cOA==
      </wsse:BinarySecurityToken>
        <ds:Signature xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
          <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='#electric-id-7E3D576B-BC09-9127-5FAC-
6CD4B194C60F'>
              <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>cx0x26UlIkT3z8q0Xiyj34bcYpA=</ds:DigestValue>
            </ds:Reference>
          </ds:SignedInfo>

<ds:SignatureValue>IML0h0QWFhYND1hbt3uD8qMFDIr16xhTYzUo2ThoCyELTpS+7OJ1jn
ZOpymLT8CIOba2+0Ua4PawpU2dwYWB02MMO89635jVqjjYzYJ+svLIzxROEhXqIByuT4HEquB
boHHTxi/lujPSNbIxbZP2rXOBF0qXJlgnUw7xqqN4N1c=</ds:SignatureValue>
          <ds:KeyInfo>
            <wsse:SecurityTokenReference>
              <wsse:Reference URI='#electric-id-B428A2B2-DA90-C072-F2BF-
F2DBF412D5FF'/>
            </wsse:SecurityTokenReference>
          </ds:KeyInfo>
        </ds:Signature>
      </wsse:Security>
    </soap:Header>
```

```
    <soap:Body
soap:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'
xmlns:wsu='http://schemas.xmlsoap.org/ws/2002/07/utility'
wsu:Id='electric-id-7E3D576B-BC09-9127-5FAC-6CD4B194C60F'>
      <n:getIntegersResponse xmlns:n='x'>
        <Result soapenc:arrayType='xsd:int[5]'>
          <i>0</i>
          <i>1</i>
          <i>2</i>
          <i>3</i>
          <i>4</i>
        </Result>
      </n:getIntegersResponse>
    </soap:Body>
  </soap:Envelope>
```

Referencias bibliográficas

[1] H.M. Deitel and et al., Web Services : A technical introduction, Editorial: Prentice Hall, Primera Edición, 2003, ISBN 0130461350

[2] Cauldwell P.; Charla R.; "La evolución de los servicios web". En Servicios Web XML Editorial Wrox Press Inc. Primera Edición. Páginas: 44 -50, 2001, ISBN 8441513635

[3] Eric Armstrong, Stephanie Bodoff, Debbie Carson, Maydene Fisher, Scott Fordin, Dale Green, Kim Haase, Eric Jendrock, "Java Web Services Tutorial", Sun Microsystems, Inc., 2003

[4] W3C , World Wide Web Consortium, Extensible Markup Languaje (XML) 1.0 http://www.w3.org/TR/1998/REC-xml-19980210, Febrero 1998

[5] The Brtitish Computer Society, A glossary of computing terms, Editorial Addison Wesley Longman, Novena Edición, 1998, ISBN 0582369673

[6] W3C, SOAP Version 1.2, SOAP Version 1.2 Part 0: Primer, http://www.w3.org/TR/soap12-part0/, Junio 2003

[7] M. Bartel,J. Boyer, B. Fox, B. LaMacchia, and E. Simon, "XML-Signature Syntax and Processing", World Wide Web Consortium, http://www.w3.org/TR/xmldsig-core/, 12 Febrero 2002

[8] A. Brown, B. Fox, S. Hada, B. LaMacchia, and H. Maruyama, "SOAP Security Extensions: Digital Signature", http://www.w3.org/TR/SOAP-dsig/, 6 Febrero 2002

[9] T. Imamura, B. Dillaway, and E. Simon, XML Encryption Syntax and Processing, http://www.w3.org/TR/xmlenc-core/, 10 Diciembre 2002

[10] OASIS, Web Services Security: SOAP Message Security 1.0 (WS-Security 2004), http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf, Marzo 2004

[11] W3C , World Wide Web Consortium, http://www.w3.org/, Noviembre 2003

[12] Mundy, D., Chadwick, D.W., An XML alternative for performance and security: ASN.1, IT Professional , Volume: 6 , Issue: 1 , Enero-Febrero. 2004, Páginas:30 – 36

[13] IBM, WebSphere Application Server – Performance, http://www-306.ibm.com/software/webservers/appserv/benchmark3.html, Noviembre 2002

[14] Vinay Bansal y Piyush Shivam, "SWSA: A Scalable Web Services Architecture", Department of Computer Sciencie, Duke University, Durham, Diciembre 2002

[15] Priya Dhawan, Performance Comparision: .NET Remoting vs. ASP.NET Web Services, Microsoft Corp., http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/bdadotnetarch14.asp, Septiembre 2002

[16] Menasce, D.A., QoS issues in Web Services,
Internet Computing, IEEE , Volume: 6 , Issue: 6 , Nov.-Dic. 2002, Páginas:72 – 75

[17] Dan Davis, Manish P. Parashar, Latency performance of SOAP implementations,
Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, Mayo 2002

[18] Takeshi Imamura, Andy Clark, Hiroshi Maruyama,
A stream-based implementation of XML Encryption
Proceedings of the 2002 ACM workshop on XML security, Noviembre 2002

[19] Deborah Russell, G.T. Gangemi, Computer security basics.
O'Reilly and Asociates, 1992, ISBN 0937175714

[20] Satoshi Hada y Hiroshi Maruyama, SOAP Security Extensions, IBM,
http://www.trl.ibm.com/projects/xml/soap/wp/wp.html, Noviembre 2000

[21] Ben Galbraith, Professional Web Services Security.
Wrox Press, Primera Edición, 2002, ISBN 1861007655

[22] Joanne Martin et al., Web Services: Promises and Compromises,
Queue, Volume 1 Issue 1, Marzo 2003

[23] N. Freed & N. Borenstein.
RFC 2045. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. http://www.ietf.org/rfc/rfc2045.txt., Noviembre 1996

[24] D. Megginson, et al., SAX: The Simple API for XML,
http://www.megginson.com/SAX/index.html, Mayo 1998.

[25] Freeman, W.; Miller, E.,
An experimental analysis of cryptographic overhead in performance-critical systems
Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 1999.
Proceedings. 7th International Symposium on , 24-28 Octubre 1999, Páginas:348 – 357

[26] W3C, XML Information Set, http://www.w3.org/TR/xml-infoset/, Febrero 2004

[27] Mollin, Richard A., An Introduction To Cryptography.
Chapman & Hall/CRC, 2001, ISBN 1584881275

[28] Ferguson, Niels , Practical Cryptography
Wiley Publishing Inc., 2003, ISBN 0471223573

[29] Diffie, W.; Hellman, M.; New directions in cryptography.
Information Theory, IEEE Transactions on , Volume: 22 , Issue: 6 , Nov. 1976
Páginas:644 – 654

[30] The RIPEMD-160 page, Hans Dobbertin, Antoon Bosselaers, Bart Preneel,
http://www.esat.kuleuven.ac.be/~bosselae/ripemd160.html, 27 Febrero 2004

[31] Secure Hash Standard, Federal Information Processing Standards, http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf, Agosto 2002

[32] The MD5 Message-Digest Algorithm, Ronald Rivest, http://www.faqs.org/rfcs/rfc1321.html, Abril 1992

[33] Dournaee, Blake, XML Security. McGraw-Hill, 2002, ISBN 0072193999

[34] W3C , World Wide Web Consortium, Canonical XML 1.0 http://www.w3.org/TR/2001/REC-xml-c14n-20010315, Marzo 2001

[35] Apache Software Foundation, WebServices – Axis, http://ws.apache.org/axis/, Julio 2004

[36] Apache Software Foundation, XML Security, http://xml.apache.org/security/index.html, Abril 2004

[37] webMethods, Glue, http://www.themindelectric.com/Solutions/Glue, Julio 2004

[38] Infomosaic, Infomosaic: The easy to use digital signature, http://www.infomosaic.net, Mayo 2004

[39] Halliden, P.W., Security for distributed applications**,** Security and Detection, 1995., European Convention on , 16-18 Mayo 1995, Páginas:156 – 160

[40] Martijn Koster, NEXOR, Robots in the Web: threat or treat?, http://www.robotstxt.org/wc/threat-or-treat.html**,** Abril 1995

[41] IBM, XML Security Suite (XSS), http://www.alphaworks.ibm.com/tech/xmlsecuritysuite, 6 Agosto 2004

[42] William F. Christopher y Carl G. Thor, Handbook for productivity measurement and improvement Productivity Press, 1993

[43] Menasce, D.A. Load testing of Web sites Internet Computing, IEEE , Volume: 6 , Issue: 4 , Julio-Agosto 2002, Páginas:70 – 74

[44] Veeramani, R.; Talbert, N., Where are we in global E-Commerce? IT Professional , Volume: 1 , Issue: 6 , Nov.-Dic. 1999, Páginas:46 – 52

[45] Tsenov, M., Soap/XML method used for data exchange between distributed databases, Intelligent Systems, 2004. Proceedings. 2004 2nd International IEEE Conference, Volume: 3, 22-24 Junio 2004, Páginas:119 – 122, Vol.3