INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS MONTERREY

PROGRAMA DE GRADUADOS EN MECATRÓNICA Y TECNOLOGÍAS DE INFORMACIÓN

TECNOLÓGICO DE MONTERREY®

EVALUACIÓN DE LA ENCRIPTACIÓN DE DATOS CONTRA LA ENCRIPTACIÓN DE ELEMENTOS SOBRE DOCUMENTOS XML

T E S I S

PRESENTADA COMO REQUISITO PARCIAL PARA OBTENER EL GRADO ACADEMICO DE:

MAESTRÍA EN CIENCIAS EN TECNOLOGÍA INFORMÁTICA

POR:

FRANCISCO GUADALUPE MACEDO JAIME

MONTERREY, N.L. **MARZO 2008**

INSTITUTO TECNOLÓGICO DE ESTUDIOS SUPERIORES DE MONTERREY

DIVISIÓN DE MECATRÓNICA Y TECNOLOGÍAS DE INFORMACIÓN

PROGRAMAS DE POSGRADO EN ELECTRÓNICA, COMPUTACIÓN, INFORMACIÓN Y COMUNICACIONES

Los miembros del comité de tesis recomendamos que la presente tesis del **Ing. Francisco Guadalupe Macedo Jaime** sea aceptada como requisito parcial para obtener el grado académico de Maestro en Ciencias de Tecnología Informática.

Comité de tesis:

_____

Alejandro Parra Briones
Asesor

_____

Dr. José Raúl Pérez Cazares
Sinodal

_____

Dr. Guillermo Jiménez Pérez
Sinodal

_____

Dr. Graciano Dieck Assad
Director del Programa de Graduados en Mecatrónica y Tecnologías de Información

MARZO 2008

EVALUACIÓN DE LA ENCRIPTACIÓN DE DATOS CONTRA LA ENCRIPTACIÓN DE ELEMENTOS SOBRE DOCUMENTOS XML

POR

FRANCISCO GUADALUPE MACEDO JAIME

TESIS

Presentada al Programa de Graduados en Mecatrónica y Tecnologías de Información.

Este trabajo es requisito parcial para obtener el título de:

Maestro en Ciencias en Tecnología Informática

INSTITUTO TECNOLÓGICO DE ESTUDIOS SUPERIORES DE MONTERREY

MONTERREY, N.L. **MARZO 2008**

Dedicatoria

A mi Dios

Por ser el soporte en los momentos mas difíciles,
por ser el guía de mis pasos por la vida,
por ser mi amigo y confidente fiel,
por ser mi LUZ, PAZ y AMOR.

A mis padres

Por inculcarme TODOS los valores que hoy viven en mí,
por enseñarme a enfrentar la vida con decisión y valor,
por apoyarme en todos mis proyectos de vida,
y sobre todo, por ser mis héroes.

A mis hermanos

Por enseñarme que aun con diferencias,
la vida es mas sencilla trabajando en equipo como familia.

A mis amigos

Por estar siempre presentes a cada momento de mi caminar,
por saber que aun sin tener la misma sangre podemos ser hermanos.

Resumen

Dada la tendencia tecnológica que presentan las empresas, al utilizar la infraestructura de Web Service para realizar sus transacciones electrónicas con sus clientes y socios, se aplican métodos de encriptación de información para garantizar la integridad, consistencia y confidencialidad de los datos. La W3C(World Wide Web Consortium) propone utilizar un método en el cual se cifran los elementos de los documentos XML(los cuales incluyen la información a ser transmitida), pero existe otra característica que es importante proveer en el servicio, la rapidez con la que la información es transmitida, por lo que surgió la inquietud de evaluar la encriptación solo de los datos a ser transmitidos a nivel aplicación, con el fin de probar cual de estas dos alternativas (encriptación de elementos XML y encriptación solo de datos) proporciona tanto seguridad como agilizar el servicio. Por lo que se creo un Web Service con la herramienta gSOAP, en el cual, se aplicaron tres algoritmos criptográficos (ElGamal, AES-128 y At-Bash) para la encriptación solo de datos; y siguiendo la recomendación de la W3C se aplicó sobre este mismo Web Service el metodo AES-128 para la encriptación de elementos de los documentos XML.

Dando como resultado la afirmación de la hipótesis propuesta para este trabajo de investigación: La encriptación de datos con algoritmos criptográficos a nivel aplicación consume un menor tiempo de ejecución que la encriptación de elementos recomendada por el estándar de la W3C.

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

Capítulo 1

Introducción

1.1 Antecedentes

La informática se inició con programas monousuarios implantados en grandes ordenadores, posteriormente estas primeras aplicaciones alcanzaron la capacidad de atender a diferentes usuarios, por lo que los Web Services[1] son parte de la revolución informática de la nueva generación de aplicaciones que trabajan colaborativamente, en las cuales el software esta distribuido en diferentes servidores, en los que un ordenador ya no se considerara como un núcleo de computo sino como un repositorio de servicios de *n* aplicaciones distribuidas por Internet y gracias a esto, el uso del software se hace mas reutilizable ya que diferentes servicios pueden ser utilizados como componentes en una aplicación y desde diferentes lugares. La estructura de programación en este tipo de ambientes está bien definida, lo que permite que cualquier empresa o persona construya, en tiempos bastante cortos, una aplicación que se comunique sin problemas con la de otra empresa.

La información que es obtenida, procesada y enviada entre los Web Services y sus clientes esta contenida en documentos XML (Extensible Markup Language), por lo cual es sumamente importante brindar seguridad sobre este tipo de documentos. Y por supuesto al ser servicios que requieren el menor tiempo de respuesta, se debe de encontrar la técnica que proporcione tanto seguridad como rapidez en el proceso del servicio.

Derivado de lo anterior nace la inquietud por conocer que técnica o técnicas son las mas adecuadas utilizar por los desarrolladores de Web Services. Por lo que este trabajo se enfoca en comparar y contrastar técnicas de encriptación, tanto de los datos como los elementos que componen los documentos XML, para contribuir con una investigación profunda de los métodos que son más viables implementar para tener un buen nivel de seguridad a nivel aplicación en Web Services al proveer servicios a dispositivos tradicionales de computo.

1.2 Objetivo

Esta tesis se enfoca a comparar y contrastar la encriptación de elementos en documentos XML, siguiendo las recomendaciones del estándar de la W3C contra la encriptación de datos a nivel aplicación utilizando tres algoritmos criptográficos. Con el objetivo de mostrar cual de las dos técnicas agiliza el servicio al aplicarle un cierto nivel de seguridad (Consistencia, integridad y confidencialidad) a la información, basándonos en la evaluación de tiempos de ejecución para las dos técnicas de encriptación (datos y elementos), se espera que la encriptación de datos sea la opción más viable para minimizar el tiempo de ejecución del servicio.

Los algoritmos criptográficos utilizados para la encriptación de datos a nivel aplicación son: ElGamal, AES-128 y At-Bash. Y para realizar la encriptación de elementos en XML se utilizaron librerías con el algoritmo AES de 128 bits.

1.3 Definición del problema

La tendencia del mercado computacional es utilizar Web Services en las empresas para poder ofrecer a sus clientes y socios un mejor servicio al poner a su disposición aplicaciones que se puedan conectar a las suyas, obteniendo un mejor desempeño en el servicio y haciendo uso de esta nueva tecnología.

En el punto 1.2, mencionamos que se desea encontrar el o los métodos más viables de encriptación para XML sobre Web Services, que ofrezcan agilizar el tiempo en el servicio, así como algunos aspectos de seguridad de la información (confidencialidad, integridad y consistencia) que están involucrados en esta tecnología, al aplicarse estos dos métodos de encriptación. Para alcanzar este nivel de seguridad, la W3C propone la encriptación de los elementos de los que están compuestos los documentos XML, aunque al realizar esta investigación no se han encontrado registros de tiempos de ejecución al utilizar este estándar, por lo que se observa la oportunidad de contribuir al evaluar este estándar y compararlo contra la encriptación de los datos contenidos en los elementos, con el fin de buscar cual es la opción mas viable utilizar, ya que hoy en día los servicios de computo deben brindar dos cuestiones fundamentales: Seguridad de la Información y agilizar las transacciones electrónicas.

Para validar nuestra propuesta se creará un servicio simple, el cual cuente con todas las características necesarias para poder ser considerado como un Web Services, el cual será ejecutado utilizando un servicio propio y utilizando clientes que accedan a este servicio. Lo anterior para poder registrar tiempos de computo, de cada petición y respuesta.

1.4 Hipótesis

La encriptación de datos con algoritmos criptográficos consume un menor tiempo de ejecución que la encriptación de elementos recomendada por el estándar de la W3C, esto, aplicado en una infraestructura de Web Services.

1.5 Alcance

Proporcionar a los desarrolladores de Web Services e investigadores, una visión más clara de dos opciones que existen para la encriptación de documentos XML (ya sea: los elementos o los datos), mediante la comparación de los métodos de encriptación anteriormente descritos, mostrando los resultados mediante algunos parámetros como: Tamaño del documento antes y después de ser cifrados, tiempos de ejecución de cada uno de los tres algoritmos criptográficos comparados con la encriptación de elementos según la W3C.

Las pruebas se llevaran acabo utilizando un documento XML con una aplicación en especifico (autorización de una compra electrónica), para fines de esta tesis se asume que los clientes son correctamente autenticados anteriormente al acceso a los documentos encriptados y que la transmisión de datos entre los servidores esta completamente asegurada por algún método de envío de datos.

A continuación se describen las herramientas que intervienen para el desarrollo de este trabajo de investigación:

Especificaciones de hardware de la computadora utilizada como servidor:

- ✓ 1 Procesador Intel celeron 1.7 GHz.
- ✓ 128 MB en RAM.
- ✓ 1 Tarjeta 10/100 Fast Ethernet.
- ✓ 30 GB de Disco duro.

Nota: Para efectos de este trabajo, por lo tanto no se realizó el análisis de las capacidades con diferentes velocidades o tipos de memoria RAM.

Especificaciones de software:

- ✓ Sistema Operativo Linux Red Hat v8.0.
- ✓ Lenguaje de programación C++.
- ✓ Compilador gcc.
- ✓ Herramienta gSOAP.

1.6 Justificación

Debido a la tendencia de las empresas a utilizar Web Services para brindar y recibir servicios de socios y/o clientes, es muy importante el aspecto de seguridad de la información y agilidad en el servicio, tanto del lado del cliente como del proveedor del servicio, debido a que cada mensaje que se intercambia (a través de SOAP) realiza múltiples saltos y se enrutan a través de numerosos puntos hasta alcanzar su destino. Por lo cual en los Web Services se ha pensado en la posibilidad de asegurar el propio mensaje desde el mismo origen (a nivel de aplicación), añadiéndose este mecanismo de seguridad a los que se apliquen en niveles inferiores, como la encriptación del documento completo, la utilización de SSL en la transmisión de paquetes, etc.

"XML Encryption es una recomendación de la W3C. De esta recomendación se desprende una funcionalidad, la cual dice: porciones de elementos de un documento XML pueden ser cifradas selectivamente" [4]. El autor realiza la siguiente aseveración: "Nótese que XML Encryption no es un nuevo tipo de encriptación. Puesto que para la encriptación de documentos XML se pueden

utilizar algoritmos criptográficos confiables y probados como: DES (Data Encryption Standard), Triple-DES (Triple-Data Encryption Standard), AES (Advanced Encryption Standard). Esto no es solo una manera de encriptación de documentos XML, porque la encriptación de un tipo de documento digital completo, cualquiera que este sea (.doc, .xls, .pdf, etc) y un documento XML no tiene diferencia. La *Encriptación Selectiva* de un documento XML es algo nuevo, lo cual no se había hecho antes de la utilización del concepto de XML Encryption" [4].

El cifrado de datos a nivel aplicación es una opción factible para garantizar la integridad, consistencia y confidencialidad de datos, para esto es necesario conocer cual o cuales algoritmos criptográficos son los más viables utilizar para este propósito y que a su vez minimicen el consumo de tiempo en su ejecución, con lo cual el servicio no se vea afectado en su rapidez.

Existe una gran variedad de algoritmos criptográficos, tanto de llave simétrica como asimétrica, por lo cual es necesario seleccionar los algoritmos que puedan ser tomados como base tanto su nivel de implementación, nivel de seguridad que proporcionan y que sean representativos de los dos grupos de algoritmos criptográficos que existen (simétricos y asimétricos).

Por lo tanto, la justificación de este trabajo de investigación se basa en hacer un juicio documentado de la opción de encriptación que es más viable para ser aplicada en una infraestructura de Web Services, así como proporcionar los resultados de la evaluación y comparación de los tres algoritmos criptográficos de los datos del documento XML contra la encriptación de elementos de un documento XML siguiendo las recomendaciones del estándar de la W3C.

1.7 Organización de la tesis

En el Capítulo 2, se introducen conceptos acerca de seguridad y cifrado de la información, los cuales serán aplicados durante el desarrollo de esta tesis. Así como una breve explicación de SSL y el por qué se afirma que para efectos de este trabajo no es necesario utilizarse. Dentro de este Capítulo también se explicaran de manera general como es que trabajan y pueden ser implementadas las técnicas de encriptación de documentos XML que se evaluaran en este trabajo.

En el Capítulo 3, se introducen y describen conceptos acerca de Web Services, se explica la diferencia que existe entre los Web Services y los Web Servers. También se encuentran detallados los estándares (UDDI, XML, WSDL y SOAP) de diseño y operación que se utilizan sobre los Web Services. Y al final de este apartado se muestra un ejemplo general del funcionamiento de un Web Services utilizando los estándares antes mencionados, en el cual se puede observar claramente cual es el flujo de la información.

En el Capítulo 4, describe gSOAP, la cual es una herramienta compuesta de librerías basadas en C y C++, fue creada con el fin de facilitar la implementación de Web Services partiendo de archivos cabecera generados por el desarrollador y obviamente de los archivos fuente para la generación tanto del cliente como del servidor del servicio, gSOAP genera algunos archivos que son utilizados como plantillas de intercambio de mensajes SOAP, evitando al programador preocuparse por estas cuestiones técnicas de la implementación. Esta herramienta es de dominio publico y fue desarrollada en The Florida State University, E.E.U.U.

En el Capítulo 5, se muestra el modelo propuesto, el cual se utilizará para realizar las implementaciones necesarias al generar y ejecutar un Web Services, cumpliendo con los estándares propios para esta tecnología. Así como también se presenta el análisis del flujo de datos que será factible aplicar para obtener resultados verificables.

En el Capítulo 6, se encuentra la descripción de las implementaciones aplicadas directamente a este trabajo de investigación, el cómo es generado el Servicio en especifico, sobre el cual se aplican los algoritmos criptográficos y se realizarán las evaluaciones necesarias, para llevar a la conclusión de este trabajo.

En el Capítulo 7, es donde se ponen en practica todos los conceptos que se mostraron en los capítulos anteriores, así como la descripción de datos, archivos y parámetros utilizados en los algoritmos criptográficos. Y por ultimo se muestran los datos y graficas resultantes de la implementación de las técnicas de encriptación sobre el Web Services que se desarrolló.

En el Capítulo 8, se muestran las conclusiones de este trabajo de investigación, basándose en los resultados obtenidos en el Capítulo 7. Lo anterior lleva a finalizar este trabajo con las aportaciones adecuadas y previstas en Capítulo 1.

En el Capítulo 9, se muestran algunos trabajos futuros, que en base a lo investigado en este trabajo de investigación surgen, con la intención de enriquecer esta área de seguridad utilizando la infraestructura de Web Services.

Capítulo 2

Seguridad y Cifrado

2.1 Introducción

El desarrollo de las telecomunicaciones ha permitido el acceso a Internet prácticamente desde cualquier lugar de mundo, por lo que en los sitios que son públicos pueden existir visitantes y personas con privilegios, y lo anterior implica proveer de seguridad en un ambiente de Web Services, el termino *seguridad* en este contexto se puede interpretar como el conjunto de técnicas que proporcionan una confiabilidad de la información, así como de los servicios, tal que puedan ser accesados solo por la persona autorizada.

2.2 Seguridad de la información

A continuación se muestran los aspectos que se deben cumplir para implementar la seguridad en un nivel general, en base a [4]:

- ✓ **Confidencialidad**: Se refiere a la protección de la información contra lectura o copiado para cualquier persona que no ha sido autorizada explícitamente por el propietario de esa información.
- ✓ **Integridad de datos:** Protección de la información (incluyendo programas) al intentar borrarla o alterarla de cualquier manera sin tener el permiso del propietario.
- ✓ **Disponibilidad:** Protección de los servicios de tal manera que no se degraden o no se encuentren disponibles. Si el servicio falla cuando un usuario autorizado intenta acceder a su información, el resultado puede ser que no pueda ver su información o se le niegue el acceso.
- ✓ **Consistencia:** Se debe cerciorar que el servicio se comporte según lo esperado para los usuarios autorizados. Si el software o hardware repentinamente se comienza a comportar radicalmente de diferente manera

a la que se comportaba, especialmente después de una actualización o solución de un problema, podría ocurrir un desastre.

- ✓ **Control:** Es la regulación de accesos para el servicio. Si logran ingresar individuos (o software) desconocidos y no autorizados en el servicio, estos pueden generar grandes problemas.

Un punto muy importe es la prioridad que el proveedor de servicios de Internet tiene de brindar seguridad y privacidad de los datos sensitivos de los clientes (Números de tarjetas, pines de seguridad, etc.). Si los compradores se sienten protegidos al realizar las transacciones, seguramente las realizaran y utilizarán el servicio frecuentemente. En el tema de seguridad tan importante es el cliente como el proveedor del servicio. Por lo que a continuación se muestran las responsabilidades que tiene el proveedor hacia el cliente.

1. Proveerle seguridad a sus clientes significa mostrar credenciales de legalidad (el proveedor del servicio debe estar legalmente constituido bajo el amparo de una empresa real) y proteger la información que viaja desde la computadora del cliente hasta su servidor en Internet (viajando desde redes de cualquier parte del mundo). Ambos requerimientos se cumplen al incluir un Certificado Digital a su dominio y la razón es la siguiente: Para obtener un certificado de este tipo, la empresa debe presentar ante organismos internacionales dedicados a tal fin (como Verisign, Thawte, etc.), los registros que la avalan como un negocio legalmente establecido. Una vez emitido el certificado, éste será instalado por NetHosting con un "componente especial" llamado clave de encriptación que le permitirá intercambiar datos con sus clientes de manera codificada que solamente podrá ser descifrada por su proveedor de servicio.

2. Si el proveedor procesa datos sensitivos de los clientes y sus bases de datos contienen información privada, se hace necesario protegerlas de accesos indeseados a través de Internet, así como implementar algoritmos criptográficos que aseguren al máximo la información, claro que este cifrado debe de consumir el menor tiempo posible en la ejecución, con el fin de ofrecerle al cliente un servicio seguro y a su vez agilizar sus transacciones.

Técnicamente es posible operar un Web Services sin componentes digitales de este tipo, pero sin ellos no se les podría dar seguridad a los clientes ni implementar un punto de venta electrónico ya que los bancos van a requerir de su instalación.

2.3 SSL

Un protocolo que se utiliza para lograr una autenticación en servidores, codificación de datos y la integridad de los mensajes es el SSL (Secure Sockets Layer). Con SSL tanto en el cliente como en el servidor, las comunicaciones en Internet serán transmitidas en formato codificado. De esta manera, puede confiar en que la información que envíe llegará de manera privada y no adulterada al servidor que se especifique.

Los servidores seguros suministran la autenticación del servidor empleando certificados digitales firmados, emitidos por organizaciones llamadas "Autoridades del certificado". Un certificado digital verifica la conexión entre la clave de un servidor público y la identificación del servidor. Las verificaciones criptográficas, mediante firmas digitales, garantizan que la información dentro del certificado sea de confianza. [2]

Un aspecto importante es, que dada la naturaleza de los accesos a los Web Services, en la actualidad existen dos grandes ambientes de cómputo: Móvil y tradicional.

- ✓ En el ambiente móvil se agrupan los dispositivos que pueden tener conexión a Internet desde cualquier lugar, aun estando en movimiento, algunos de ellos son: PDA's (Personal Digital Assistant) y teléfonos celulares.
- ✓ En el ambiente tradicional se encuentran los dispositivos que están instalados de manera un tanto estática, dentro de esta categoría se pueden mencionar algunos como: PC's, Laptop's y Server´s.

Ahora bien, este protocolo esta orientado a la conexión o transporte de datos, por lo cual se puede tomar como una alternativa viable para arquitecturas de servicios que no cuentan con ningún mecanismo de aseguramiento (cifrado) de datos o que se quieren reforzar los mismos, aunque se tiene que tomar en cuenta que si para el tipo de servicio brindado la rapidez de respuesta es un factor clave, el utilizar este protocolo disminuirá este rubro.

"La importancia principal de la encriptación sobre XML para Web Services es, que permite el principio de seguridad de confidencialidad para ser satisfecha en mas que solo el contexto de una simple petición SOAP".[4]

"Un escenario obvio es el cual, si la información en un mensaje SOAP debe permanecer confidencial mientras este es enviado sobre una transacción multipunto. En este escenario, si solo es utilizado SSL, existirá un hueco en cada

punto final de SOAP, donde el dato sensitivo podría estar temporalmente vulnerable"[4].

Por lo tanto, para fines de este trabajo de investigación el protocolo SSL se dejara a un lado, soportando esta decisión en que lo que se pretende es encriptar solo porciones de datos que utilizaran los documentos XML y no todo el documento como tal, para no afectar al desempeño del servicio. Aunque como se menciona en el párrafo anterior, si lo que se busca es mayor seguridad y no tanto velocidad en el servicio, no seria una mala idea el utilizar ambas opciones (encriptar datos sobre XML y utilizar el protocolo SSL).

2.4 Cifrado y Descifrado

Una parte muy importante dentro de seguridad computacional es la criptografía que se utiliza, por lo que es necesario conocer acerca de este tema para entender mejor el concepto de seguridad de manera general.

La criptografía se define como la técnica de ocultación de información mediante la codificación y decodificación de contenidos, el término proviene del griego "kruptos" que significa "oculto".[3]

La encriptación o cifrado consiste en aplicar una serie de operaciones matemáticas (algoritmo) a un texto legible, para convertirlo en algo totalmente inteligible. Para que el programa informático que aplica este algoritmo funcione, necesita el texto a codificar y una *clave de usuario*, de manera que distintas claves aplicadas a un mismo texto original, den lugar a resultados diferentes y únicos para cada una. Conocida la clave con la que se codificó, se podrá aplicar el algoritmo de descodificación al texto cifrado para obtener el original. De esta manera, sólo se tiene acceso a la información si se dispone de la clave de usuario correcta, aunque es importante destacar que no se refiere al uso de las cadenas que definen al usuario o contraseña.

El descifrado, se basa en una misma clave para cifrar y descifrar el texto. El usuario que pretende ocultar el contenido de un texto personal, no tendrá mayores inconvenientes, ya que, tras la codificación, el texto queda a salvo de miradas curiosas, y solo él conoce la clave de acceso a la información que contiene. El problema surge cuando se pretenden intercambiar textos cifrados (por ejemplo, envíos de información confidencial). Cuando una misma clave está en poder de cierto número de personas, las posibilidades de que caiga en manos de alguien ajeno al grupo se multiplican, teniendo, de esta manera, acceso a toda la información, lo que supone un gran fallo de seguridad.

A continuación, en la tabla 2.1 se muestra un Benchmark de los algoritmos criptográficos mas reconocidos, con el fin de dar una idea de la velocidad y procesamiento que pueden llegar realizar en un determinado tiempo.

Todos los tiempos fueron registrados después de realizar la codificación de los algoritmos en C++, compilados con Microsoft Visual C++ .NET 2003 (la optimización para su rapidez esta considerada en la generación P4) y fueron ejecutados en un procesador Pentium 4 a 2.1 GHz bajo un Windows XP SP 1.386.[38]

Algoritmo	Bytes Procesados	Tiempo ejecución	Megabytes(2^20 bytes)/Segundo
SEAL-3.0-BE	2147483648	2.443	838.314
SEAL-3.0-LE	2147483648	2.224	920.863
Panama Cipher (little endian)	2147483648	2.093	978.5
Adler-32	2147483648	1.772	1155.756
Panama Cipher (big endian)	1073741824	1.382	740.955
MD5	536870912	2.503	204.555
HMAC/MD5	536870912	2.373	215.761
Panama Hash (big endian)	536870912	2.333	219.46
Panama Hash (little endian)	536870912	1.693	302.422
CRC-32	536870912	1.572	325.7
WAKE-CFB-BE	268435456	2.633	97.227
ARC4	268435456	2.313	110.679
HAVAL (pass=3)	268435456	2.164	118.299
WAKE-CFB-LE	268435456	1.923	133.125
WAKE-OFB-BE	268435456	1.913	133.821
WAKE-OFB-LE	268435456	1.792	142.857
XMACC/MD5	268435456	1.673	153.019
MD5-MAC	268435456	1.372	186.589
SHA-256	134217728	2.623	48.799
Rijndael (256-bit key)	134217728	2.604	49.155
RIPE-MD160	134217728	2.533	50.533
MDC/MD5	134217728	2.434	52.588
Rijndael (192-bit key)	134217728	2.283	56.067
Rijndael (128) OFB	134217728	2.213	57.84
CBC-MAC/Rijndael	134217728	2.203	58.103
Rijndael (128) CTR	134217728	2.173	58.905
RC5 (r=16)	134217728	2.163	59.177
DMAC/Rijndael	134217728	2.113	60.577

Rijndael (128-bit key)	134217728	2.063	62.046
Blowfish	134217728	2.003	63.904
HAVAL (pass=5)	134217728	1.772	72.235
SHA-1	134217728	1.763	72.604
HAVAL (pass=4)	134217728	1.612	79.404
TEA	67108864	2.554	25.059
MARS	67108864	2.434	26.294
Square	67108864	2.053	31.174
Twofish	67108864	2.012	31.809
RC6	67108864	1.922	33.299
SHARK (r=6)	67108864	1.843	34.726
GOST	67108864	1.732	36.952
Rijndael (128) CFB	67108864	1.622	39.457
Tiger	67108864	1.613	39.678
CAST-128	67108864	1.472	43.478
Rijndael (128) CBC	67108864	1.423	44.975
3-WAY	66846720	1.813	35.163
Luby-Rackoff/MD5	33554432	2.533	12.633
IDEA	33554432	1.813	17.65
DES-XEX3	33554432	1.703	18.79
SAFER (r=8)	33554432	1.692	18.913
SKIPJACK	33554432	1.573	20.343
Serpent	33554432	1.503	21.291
CAST-256	33554432	1.452	22.039
DES	33554432	1.442	22.191
Diamond2	16777216	1.702	9.401
DES-EDE3	16777216	1.632	9.804
Diamond2 Lite	16777216	1.412	11.331
SHA-512	16777216	1.372	11.662
RC2	16777216	1.372	11.662
MD2	4194304	1.583	2.527
BlumBlumShub 512	131072	1.903	0.066
BlumBlumShub 1024	65536	2.063	0.03
BlumBlumShub 2048	16384	1.442	0.011

Tabla 2.1 Benchmark de algoritmos criptográficos [38].

Nota: Cabe mencionar que el algoritmo **Rijndael** es el **AES.**

El cifrado y descifrado de **ElGamal** utiliza exponentes muy cortos para disminuir el tiempo de ejecución. Por lo que este algoritmo no se incluyó en la tabla 2.1, ya que su desempeño varia dependiendo del exponente que se utilice. El tamaño de los exponentes secretos son elegidos de tal manera que se eviten ataques de robo y detección de claves en medio del envío, disminuyéndolo con un algoritmo de logaritmo discreto general. Los tamaños utilizados generalmente para estas llaves son:

Modulo	exponente
512	120
1024	164
2048	226

Tabla 2.2 Módulos y exponentes para ElGamal [38].

Ningún sistema de encriptación es ideal para todas las situaciones, esto depende del grado de sensitividad de los datos y la rapidez en el servicio que se desee.

La tabla 2.3 ilustra algunas de las ventajas y desventajas de cada tipo de encriptación, así como las diferencias entre los algoritmos y las longitudes de las llaves.

Los puntos que deben tomarse en cuenta para seleccionar el algoritmo criptográfico más adecuado son:

1. Determinar qué tan sensibles son los datos que se utilizarán en la transacción electrónica.
2. Por cuánto tiempo serán sensibles y necesitan estar protegidos.

Una vez que se respondan las dos cuestiones anteriores, se selecciona un algoritmo de encriptación y longitud de llave que tome más tiempo en descifrarse que el período de tiempo por el cual sus datos serán vulnerables.

Encriptación	Ventajas	Desventajas
Llave simétrica	• Rápida. • Se puede instrumentar fácilmente en hardware.	• Ambas llaves son la misma. • Dificultad para distribuir las llaves. • No soporta las firmas digitales.
Llave pública	• Usa dos llaves diferentes. • Llaves relativamente fáciles de distribuir • Proporciona integridad y no repudiación mediante firmas digitales.	• Lenta y exhaustiva.

Tabla 2.3 Diferencias entre llaves simétricas y publicas [11].

La tabla 2.4 es una condensación de la tabla de Schneier, la cual estima el costo de construir una computadora en 1995 para violar llaves simétricas y el tiempo requerido para violar ciertas longitudes de llave. Comparación de tiempo y dinero (USD) necesarios para violar distintas longitudes de llave [12].

Longitud de la llave en bits					
Costo	**40**	**56**	**64**	**80**	**128**
$100 mil	2 segs	35 hrs	1 año	70000 años	1019años
$1 millón	.2 segs	3.5 hrs	37 días	7000 años	1018años
$100 millones	2 milisegs	2 mins	9 hrs	700 años	1016años
$1 billón	.2 milisegs	13 segs	1 hr	7 años	1015años
$100 billones	2 microsegs	.1 seg	32 segs	24 días	1013años

Tabla 2.4 Esfuerzo para romper llaves [12].

El poder de cómputo siempre va en incremento y los costos descienden, así que será más fácil y barato violar llaves más largas en el futuro. Estas estimaciones son para ataques de fuerza bruta, es decir, para adivinar cada llave posible. Existen otros métodos para violar llaves, dependiendo de los códigos usados (esto es lo que mantiene ocupados a los analistas de criptografía); sin embargo, los estimados de ataques de fuerza bruta a menudo se citan como una medida de la fortaleza de un método de encriptación [11].

Longitudes de llave secreta y llave pública para niveles equivalentes de seguridad:

Longitud de la llave secreta	Longitud de la llave pública
56 bits	384 bits
64 bits	512 bits
80 bits	768 bits
112 bits	1792 bits
128 bits	2304 bits

Tabla 2.5 Equivalencias de seguridad en llaves [11].

2.4.1 Algoritmo de cifrado de ElGamal

Existen versiones sobre Criptositemas de Curvas Elípticas (por sus siglas en ingles ECC) de distintos algoritmos asimétricos, como RSA (llamado así por las siglas de sus creadores: Rivest, Shamir y Adleman) y DSA (Digest SignatureMethod's Algorithm), pero quizás el algoritmo que tiene un apoyo más evidente y directo sobre el Problema del Logaritmo Discreto (por sus siglas en ingles DLP), este problema es una función de un solo camino, basado en la dificultad de encontrar un logaritmo en un grupo de puntos cardinales. El DLP ha sido extensamente estudiado y se ha basado en algunos sistemas criptográficos de llave publica, como el ElGamal[8] .

A continuación se muestra como es que este algoritmo logra ser definido.

a) Generación de claves:

1. Tomar un valor x y un grupo finito Zn {x, x^2, x^3, ...x^n}. Sea G un valor calculado que se encuentra dentro de Zn.
2. Seleccionar un número aleatorio a, entre 1 y n-1, y calcular x^a.
3. La clave pública es el par (x,x^a). La clave privada es a.

b) Cifrado:

1. Sea m el texto en claro.
2. Seleccionar aleatoriamente k, entre 1 y n-1.
3. Calcular g=x^k y f=m(x^a)^k
4. El criptograma c es el par (g,f).

c) Descifrado:

1. Calcular g^a y luego g^(-a) (su inversa modular).
2. m=(g^-a)*f

Bastaría con sustituir en conjunto Zn por algún GCE (Grupo de Curvas Elípticas) sobre enteros módulo o sobre polinomios, y las operaciones de multiplicación por sumas en Curvas Elípticas.

"Las Curvas Elípticas son una estructura matemática, que presenta propiedades adecuadas (conjunto muy amplio de puntos cardinales, precisión matemática para encontrar el adecuado por el propietario) para su uso en Criptografía, debido a que en lugar de usar números enteros como los *símbolos* del alfabeto del mensaje a encriptar, usa puntos en un objeto matemático llamado *Curva Elíptica*. ECC puede ser usado tanto para encriptar como para firmar

digitalmente un documento. Hasta el 2002 (año en el que se publico este articulo), no se conoce ataque alguno cuyo tiempo de ejecución esperado sea sub-exponencial para poder romper los ECC, esto hace que para obtener el mismo nivel de seguridad que brindan los otros sistemas (DSA y RSA), el espacio de claves de ECC sea mucho más pequeño, lo que lo hace una tecnología adecuada para ser utilizada en ambientes restringidos en recursos (memoria, costo, velocidad, ancho de banda, etc.)". [35]

El objetivo es mostrar como las curvas elípticas construidas sobre un cuerpo finito (grupo de valores que son generados de manera finita, los cuales constituyen un GCE) son una herramienta idónea para implementar sistemas criptográficos de clave pública capaces de ofrecer mejores parámetros y/o mayor nivel de seguridad que otros sistemas clásicos (RSA, problema del logaritmo discreto sobre un cuerpo finito, etc.). El cálculo del número de puntos dentro de las curvas elípticas sobre un cuerpo finito, permite la selección de curvas elípticas que son buenas candidatas para propósitos criptográficos.

2.4.2 Algoritmo At-Bash

Este sistema tiene sus orígenes en el alfabeto judío y es conocido por el nombre de At-Bash debido a la primera letra "alef " y la última "tav", sale de aquí "At". La segunda letra se llama "beit", y la antepenúltima "shin", de aquí sale "Bash". O sea que este sistema consiste en cambiar la última letra por la primera, la anteúltima por la segunda, y así con todo el abecedario.

Con el paso de los años este algoritmo fue utilizado por diferentes culturas para realizar encriptaciones de escritos que serian enviados entre ciudades y que de ser interceptados no fueran entendidos por sus enemigos. El emperador Julio Cesar utilizó este método con un intercambio de caracteres con un desfase de tres posiciones y a este método se le conoció como At-Bash 3. Cabe mencionar que para estas implementaciones se utiliza el alfabeto nativo de las culturas que lo utilizaban.

Por ser un algoritmo simétrico, desde el inicio de la implantación se debe definir el numero de posiciones que se utilizaran para el desfase en la cadena a encriptar, o bien, si se seguirá el sistema original (intercambiar primero por ultimo, segundo por penúltimo, etc.), después de haber definido estas primicias, lo que tiene que hacerse es implementar tanto para el cifrado como el descifrado el mismo método, con lo cual se evita enviar claves privadas o publicas que puedan ser sustraídas

por personas que desean observar la información que viaja por el medio de transporte.

2.4.3 Algoritmo AES

El gobierno americano adopta el estándar de encriptación AES, el cual sustituye al algoritmo DES que venían utilizando en todas sus comunicaciones desde 1977. El Departamento de Comercio americano anunció en diciembre del 2001 su intención de realizar este cambio en el método de encriptación de todas sus comunicaciones informáticas[13].

El algoritmo AES, también conocido como Rijndael, fue desarrollado por los científicos belgas Vincent Rijmen y Joan Daemen fue desarrollado en los 90's pero no fue hasta el 2 de octubre del 2000 cuando el Instituto Nacional de Estándares y Tecnología (NIST) norteamericano lo seleccionó como el mejor estándar de encriptación entre más de una docena de competidores correspondientes a criptólogos de considerable fama mundial y a empresas de renombre como IBM, RSA o Conterpane.

El gran avance de este estándar es que además de trabajar con claves de 128 y 192 bits, también aceptan longitudes de 256 bits, una característica importante si se tiene en cuenta que DES sólo permitía claves de 56 bits. Para comparar la eficacia de AES, desde su página web NIST afirma que mientras que el desfasado DES podría actualmente quebrantarse después de varias horas de intento, se necesitarían 149 trillones de años con un algoritmo AES de sólo 128 bits siempre y cuando se crease previamente el equipo adecuado para hacerlo[13].

El nuevo estándar se puede aplicar a una amplia variedad de transacciones electrónicas: desde cajeros automáticos, comercio electrónico, e-mails confidenciales, etc. Por este motivo, el secretario de comercio Don Evans afirma que, "AES ayudará a la nación a proteger sus infraestructuras de información crítica y asegurar la privacidad de la información personal de todos los ciudadanos americanos"[13].

La tabla 2.6 muestra el procesamiento con una frecuencia de reloj de 360Mhz. En la tabla se observan los resultados de varias implementaciones de algoritmos criptográficos.

Algoritmo	Total de ciclos	Bits/Ciclo	Procesamiento (Mbps)
3DES	336 (7ciclos*48)	0.19	68
3DES corr.	392 (7ciclos*56)	0.16	59
AES-128/128	90 (9 ciclos*10)	1.42	511
AES-128/128 corr.	130 (9 ciclos*13)	0.98	353

Tabla 2.6 Desempeño del AES [14].

"Los resultados de este trabajo de investigación son basados en simulación de software y resultados de síntesis de partes de hardware dedicado. El autor esta consciente de que sus mediciones no han sido realizadas en hardware existente, y en la tabla 2.7 se tienen los resultados de simulaciones de algoritmos criptográficos"[14].

Algoritmo	Ciclos	Bits/Ciclo	Procesamiento (Mbps) a 400Mhz
AES-128/128	80	1.6	640
AES-128/128	70	1.83	732
DES	35	1.83	732
3DES	105	0.61	244
MD5	504	1.02	406
SHA-1	488	1.05	420

Tabla 2.7 Desempeño del AES [14].

2.5 Técnicas de Encriptación para XML

Confidencialidad significa que la información en el servidor y en la transmisión no pueda ser accedida por partes no autorizadas. Cuando la información se encuentra en el servidor, como en una base de datos, una política de control de acceso muy fuerte puede ser utilizada para brindar confiabilidad. Cuando la información es transmitida, la encriptación es a menudo lo mas apropiado para ofrecer confiabilidad.[4]

"Expresar los datos como XML no es tan emocionante ahora que el mundo entero esta migrando a expresar sus datos con algunas estructuras de datos en XML. De cualquier manera, las ventajas generales de XML son aplicadas. Incluir datos encriptados en XML es una ventaja para el cliente de los datos encriptados, desde un XML DOM (Document Object Model). puede ser utilizado para extraer el

texto cifrado y el documento original, esto en un algoritmo (por supuesto con una llave) para el descifrado".[4]

Dentro de un taller llamado: *Proceedings of the 2002 ACM workshop on XML security*, la ACM (Association Computing Machinery) propone dos métodos para lograr la encriptación en documentos XML, los cuales son detallados a continuación.

2.6 Resumen

La criptografía es la herramienta más poderosa para proporcionar muchos servicios de seguridad de la información. Resulta la más adecuada además para los sistemas informáticos. La criptografía es una rama de las matemáticas que se ocupa del proceso de cifrado de la información. El cifrado de datos es una técnica que permite transformar cierta información en una serie de datos ininteligibles o "datos cifrados" a veces con capacidad de recuperación (los datos pueden ser "descifrados") y a veces sin esta posibilidad.

Por lo cual es importante el elegir algoritmos criptográficos adecuados, que nos ayuden a brindar seguridad a la información y que no comprometan el desempeño del servicio, en este caso en las transacciones electrónicas.

En este capitulo se describen tres algoritmos: At-bash, ElGamal y AES, los cuales serán utilizados para realizar la encriptación de la información en un Web Service, con el fin de evaluar su desempeño en tiempo de ejecución.

En un primer acercamiento por realizar la implementación de un algoritmo de encriptación de datos, utilizando sobre un Web Service, se pensó en usar uno que fuera sencillo de programar y que sirviera como base para desarrollos posteriores, fue así que se eligió el algoritmo At-Bash, el cual, como se explico en el punto 2.4.3 es considerado como uno de los primeros algoritmos que existieron y que se utilizó por muchos años para la transmisión de información de manera segura, aunque hoy en día, con las velocidades de procesamiento con las que se cuenta, el descifrado es cada vez mas rápido.

Posteriormente, se buscaron dos algoritmos criptográficos que brinden el nivel de seguridad deseado (Consistencia, Integridad y Confidencialidad) y que a su vez fueran representativos de cada uno de los dos grupos principales de algoritmos (llave simétrica y asimétrica).

Se eligió ElGamal por ser un algoritmo de llave publica o asimétrica, el cual, como se describió en el punto 2.4.1:

"Hasta el 2002 (año en el que se publico este articulo), no se conoce ataque alguno cuyo tiempo de ejecución esperado sea sub-exponencial para poder romper los ECC, esto hace que para obtener el mismo nivel de seguridad que brindan los otros sistemas de llave publica (DSA y RSA), el espacio de claves de ECC sea mucho más pequeño, por lo que se convierte en una tecnología adecuada para ser utilizada en ambientes restringidos en recursos (memoria, costo, velocidad, ancho de banda, etc)". [35]

Y por ultimo se eligió el algoritmo AES, también conocido como Rijndael por ser un algoritmo de llave privada o simétrica, el 2 de octubre del 2000 el Instituto Nacional de Estándares y Tecnología (NIST) norteamericano lo seleccionó como el mejor estándar de encriptación entre más de una docena de competidores correspondientes a criptólogos de considerable fama mundial y a empresas de renombre como IBM, RSA o Conterpane.

"NIST afirma que mientras que el DES podría en el año 2000 quebrantarse después de varias horas de intento, se necesitarían 149 trillones de años con un algoritmo AES de sólo 128 bits siempre y cuando se crease previamente el equipo adecuado para hacerlo" [13].

Capítulo 3

Web Services

3.1 Introducción

Internet está evolucionando rápidamente de los sitios web actuales, que simplemente proporcionan páginas de interfaz de usuario a través de exploradores, a una futura generación de sitios web programables que establecen vínculos directamente con organizaciones, aplicaciones, servicios y dispositivos entre sí, los llamados Web Services.

Un Web Services de manera general, es cualquier servicio que esta disponible a través Internet, que utiliza un sistema de mensajes XML estandarizado y no esta atado a un solo sistema operativo o lenguaje de programación. Hay algunas alternativas para el envío de mensajes XML. Por ejemplo, se podría utilizar el XML Remote Procedure Calls (XML-RPC) o SOAP(Simple Objet Access Protocol).[9]

3.2 Definición de Web Services

En [5] se define un Web Services como: aquel servidor de web que provee de programas de software que utilizan XML para intercambiar información con otro software por medio de los protocolos comunes que se usan para Internet. A estos programas se le conoce como Web Services, el cual, básicamente se comunica a través la web para proveer un conjunto especifico de operaciones, de tal forma que otras aplicaciones puedan ser invocadas.

En la Figura 3.1 se muestra un ejemplo básico del funcionamiento de un Web Services al momento de recibir una petición de un cliente de manera remota.

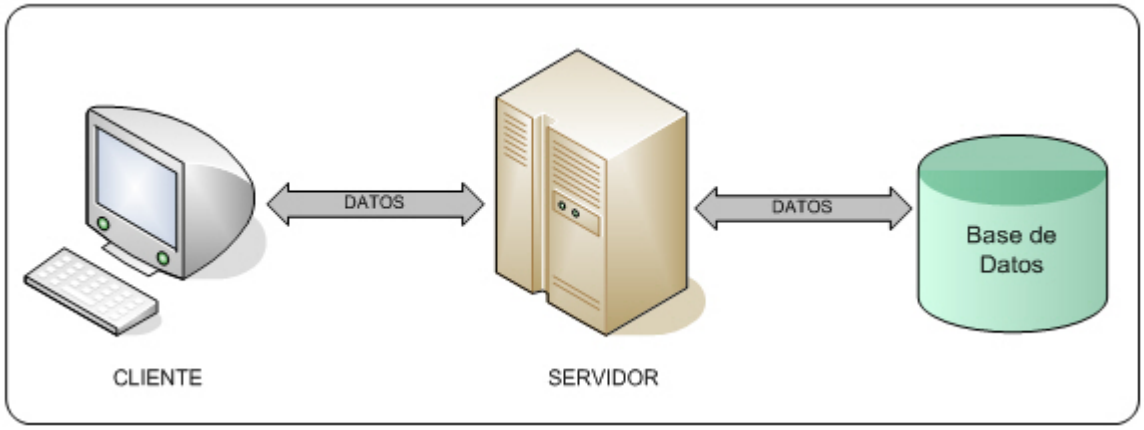


Figura 3.1 Petición y respuesta de un servicio en un Web Services.

Existen algunos beneficios que provee el utilizar Web Services sobre otras tecnologías, ya que, la infraestructura de los Web Services resuelve algunos de los problemas inherentes a las tecnologías de computo distribuido, puesto que son: modulares, se auto describen, y pueden ser invocados y publicados desde cualquier punto de la web. Los Web Services pueden ser desde simples funciones o métodos hasta complejos procesos de negocio. Y esta es la razón por la cual se cree que los Web Services es la tendencia más notable a seguir hablando de computación distribuida.[5]

Hoy en día los Web Services trabajan sobre http, por lo que las compañías puede exponerse y al acceder usando esta tecnología, con la que ellos ya cuentan. Otra ventaja es que los Web Services son mas ínter operables, por que ellos emplean un código basado en el estándar XML para comunicarse entre sistemas. Además, casi todos los vendedores importantes de software acordaron utilizar los mismos estándares, como son: SOAP, WSDL y UDDI (En el punto 3.3 se muestran a detalle).

Los Web Services tienen la capacidad de proveer una comunicación directa de aplicación a aplicación, abriendo la oportunidad de explotar el potencial (al ser modular, auto descriptible y ejecutarse sobre HTTP) del uso de la comunicación. También cuentan con el potencial de cambiar los procesos completos de diseño, desarrollo e implementación del software. En este contexto, ellos representan un nuevo paradigma de la computación distribuida.[5]

Las dos agrupaciones que desarrollan estándares para Web Services mas reconocidas son:

- ✓ **W3C (World Wide Web Consortium):** Es un consorcio industrial internacional, cuenta con más de 500 organizaciones gubernamentales, no gubernamentales e industrias. Su finalidad es promover la evolución e interoperatividad de la Web, para fomentar su universalidad.

- ✓ **OASIS (Organization for the Advancement of Structured Information Standards):** Esta organización no tiene fines de lucro, el consorcio global fue creado para conducir al desarrollo, la convergencia y la adopción de los estándares en e-bussines.

3.2.1 Diferencia entre Web Services y Web Server

Antes de los 90's ya existía Internet, pero no el WWW, entonces, se tuvo que añadir un software a los servidores de Internet que deseaban servir páginas Web, dicho software se conoce como Web Server. Un Web Server como tal, es la aplicación dedicada a publicar información (en formato HTML) en una intranet o en Internet, realizando esta tarea por medio del protocolo de transferencia de hipertexto (HTTP), la Web funciona básicamente por medio de peticiones y respuestas, es decir, el navegador solicita información enviando una dirección URL a un servidor, el cual, responde enviando una página en código HTML(HiperText Markup Language)[5].

Por lo tanto la diferencia principal de un Web Server y un Web Services es, que los primeros proveen un servicio utilizando HTML, CGIs, ASPs, etc., pero sin la interacción directa con otro software para generar un servicio, mientras que los Web Services intercambian información de un software a otro utilizando la web para proveer el servicio.

3.3 Estándares de diseño y operación

En puntos anteriores se ha mencionado que algunas agrupaciones se están preocupando por brindar estándares de diseño y operación para Web Services, estos estándares han sido aceptados por los vendedores de software (ej. Microsoft e IBM) para esta tecnología y sus desarrollos en este sentido están siendo enfocados para que sus productos los soporten y brinden la mayor Interoperabilidad.

La base de los documentos que intercambian los Web Services son: HTML y XML. El desarrollador puede crear programas accesibles desde cualquier

dispositivo que soporte estos estándares, aprovechando la conectividad de Internet. Se pueden crear servicios accesibles desde Internet. En la Figura 3.2 se muestra gráficamente en donde se encuentran ubicadas cada uno de los estándares en un esquema de Web Services.

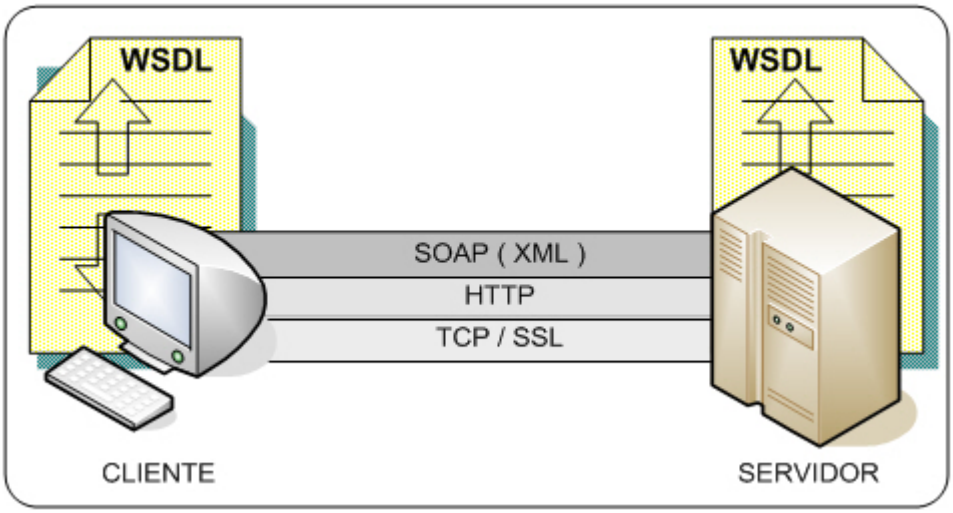


Figura 3.2 Estándares en Web Services.

3.3.1 XML

El XML(Extensible Markup Language) proviene de un lenguaje que inventó IBM en 1969. El lenguaje de IBM se llama GML (General Markup Language) y surgió por la necesidad que tenían en la empresa de almacenar grandes cantidades de información de temas diversos.

IBM al ver la tuvo la necesidad de almacenar la enorme documentación con la que contaba, sobre todas las áreas en las que trabajaba e investigaba, y la cantidad de información que día a día iría generando, por lo que busco una manera de guardar la información y los expertos de IBM inventaron GML, un lenguaje con el cual pudieran clasificarlo todo y escribir cualquier documento para que se pueda luego procesar adecuadamente. [5]

Este lenguaje gustó mucho a la gente de ISO, de esta manera en 1986 trabajaron para normalizar el lenguaje, creando el SGML, que no era más que el GML pero estándar (Standar General Markup Language). SGML es un lenguaje muy utilizado, capaz de adaptarse a un gran abanico de problemas y a partir de él se han creado los siguientes sistemas para almacenar información. [5]

En 1989, para el ámbito de la red Internet, un usuario que había conocido el lenguaje de etiquetas (Markup) y los hiperenlaces creo un nuevo lenguaje llamado HTML, que fue utilizado para un nuevo servicio de Internet, la Web. Este lenguaje fue adoptado rápidamente por la comunidad y varias organizaciones comerciales crearon sus propios visores de HTML y pelearon entre ellos para hacer el browser más avanzado, inventándose etiquetas como su propia voluntad les decía. Desde los 90s hasta hoy la W3C ha tratado de poner orden en el HTML y establecer sus reglas y etiquetas para que sea un estándar. Sin embargo el HTML creció de una manera descontrolada y no cumplió con resolver todos los problemas que planteaba la sociedad global de Internet. [5]

El mismo W3C empezó y continúa, en el desarrollo de XML. Pretendían solucionar las carencias del HTML en lo que se respecta al tratamiento de la información. Por lo cual también ha brindado el apoyo a las compañías que han desarrollado otras tecnologías que puedan dar al XML un soporte mayor, como SOAP y WSDL.

3.3.1.1 XML VS HTML

En teoría, HTML es un subconjunto de XML especializado en presentación de documentos para la Web, mientras que XML es un subconjunto de SGML especializado en gestión de información para la Web.

En la práctica, HTML está un poco dentro de XML (que a su vez es parte de SGML) y otro poco fuera de SGML.

El grupo W3C ha dictado reglas expresas para distinguir el HTML que sigue al pie de la letra las normas de XML, denominándolo XHTML (eXtendible HyperText Markup Language), que no es más que una reformulación de HTML 4 dentro de las normas de XML.

- ✓ HTML está orientado a la presentación de datos, en cambio, XML está orientado a los datos en sí mismos.

- ✓ HTML es un lenguaje de presentación, define un conjunto de etiquetas y atributos válidos, una utilización válida de estos elementos y un significado visual para cada elemento del lenguaje. Mientras que, XML es un metalenguaje (un lenguaje para definir otros lenguajes). XML no define las etiquetas ni cómo se utilizan, sólo define unas pocas reglas sintácticas para crear documentos.

- ✓ Cualquier programa informático trabajará mejor con datos en XML.
- ✓ XML no sustituye a HTML pues sirven para cosas distintas: HTML para presentar información en páginas web y XML para representar e intercambiar datos, independientemente de su presentación. XML y HTML son complementarios.

3.3.2 SOAP

SOAP (Simple Objet Access Protocol) fue creado antes de la llegada de los Web Services, como un protocolo de comunicación que pudiera ser utilizado sobre el Internet. Cuando W3C lanzo el XML 1.0 como recomendación en 1998, un grupo de desarrolladores lo realizaron en adición a una descripción de datos, XML también puede describir acciones pragmáticas o comportamientos. Inicialmente IBM, Lotus Development Corporation, Microsoft, DevelopMentor y Userland Software iniciaron a desarrollar un protocolo de mensajería para XML, para definir una manera de especificación no importando la plataforma (interoperable) para invocar operaciones remotas. Fue conceptualizado en 1998 y finalmente en 1999 se lanza el SOAP v0.9.[5]

Las organizaciones que desarrollaron SOAP publicaron la primer especificación SOAP llamándola SOAP 0.9 en 1999, se siguieron realizando mas versiones antes de someterlo a la aprobación de W3C. [5]

En Julio del 2001, la W3C lanza SOAP 1.2 como un bosquejo del trabajo realizado. Dicho en otras palabras, se publicó el documento de especificación SOAP, pero continuó aceptando comentarios y sugerencias para cambiar este documento. Como resultado, se observo que era inestable y se sujetó a cambios, por lo tanto no fue conveniente para la implementación en las grandes industrias. [5]

Otra alternativa para el intercambio de información utilizando la infraestructura de un Web Service es el XML-RPC (XML Remote Procedure Call), ya que al igual que SOAP, provee un servicio de comunicación entre cliente-servidor. Y la diferencia básica entre SOAP y XML-RPC es que los procedimientos de SOAP toman los parámetros que se encuentran en la petición y se coloca donde se encuentran en el código dentro del XML. El orden de estos parámetros es irrelevante, pero, en XML-RPC el orden es relevante y los parámetros no son utilizados. [10]

3.3.2.1 Arquitectura de SOAP

La especificación SOAP ofrece un protocolo por el cual los mensajes XML pueden comunicar de manera práctica instrucciones entre aplicaciones. Esta especificación abarca cuatro partes principales según se describe en [10]:

- ✓ La primera es empaque SOAP, el cual describe el formato de un mensaje SOAP.
- ✓ El segundo define un conjunto de reglas de *encode data types*, en otras palabras es la forma en la que la información será enviada. La mayoría de lenguajes de programación requiere de ambientes para tipos de datos específicos los cuales son utilizados por una aplicación. Estos programas permiten que se reciban los datos para interpretar la información correctamente.
- ✓ La tercera parte define como un mensaje SOAP puede ejecutar un *remote procedure call.*
- ✓ La ultima parte de SOAP se le llama SOAP binding framework, el cual define un protocolo por el cual los mensajes de SOAP son transmitidos a las aplicaciones.

3.3.2.2 Clientes y Servidores de SOAP

De acuerdo a [5], se describe que:

Una aplicación cliente es aquella que quiere realizar una consulta a un Web Services utilizando un cliente de SOAP para crear y enviar un mensaje en SOAP. La especificación SOAP no indica el tipo de aplicaciones que pueden funcionar como clientes SOAP; las aplicaciones pueden ser simples programas o parte de un sistema complejo. Antes de crear un mensaje, un cliente puede usar los registros de UDDI u otro mecanismo de descubrimiento para encontrar en un Web Services.

El Web Services registra la información en los registros de UDDI. Después, el cliente busca en estos registros y localiza el servicio deseado, el cliente crea un mensaje SOAP de acuerdo a la información encontrada en los archivos de WSDL.

Un servidor SOAP es un programa o parte de un programa que recibe y procesa las solicitudes de un cliente SOAP. El servidor espera un mensaje, cuanto este llega, lo acepta y procesa la información en código que sea entendible para el

Web Services. El servidor SOAP logra esto por medio del uso de Parsers. Un Parser es un módulo, biblioteca o programa que se ocupa de transformar un archivo de texto en una representación interna. En el caso de XML, como el formato siempre es el mismo, no necesitamos crear un parser cada vez que se desarrolla un programa, ejemplos de parsers son: SAX (Simple API for XML) y DOM.

Una vez que el cliente envía la petición al servidor con los datos contenidos en el mensaje SOAP, el servidor realiza las tareas definidas en el mensaje y envía los datos de respuesta de regreso al cliente en otro mensaje SOAP. Cada proceso de envío y recepción de este tipo de mensajes involucran una traducción de tipos de datos para ser entendibles para el Web Services, haciéndolo independiente de la plataforma que se este utilizando.

3.3.2.3 Mejoras a SOAP

De acuerdo a [5], se describe que: SOAP aun es una nueva especificación y esto no esta enfocado en todos los aspectos de la interacción con Web Services. Como la W3C y otras organizaciones desarrollan tecnología dicen que las nuevas versiones de SOAP tendrán mejoras en su funcionamiento (definición de datos, más eficiente manejo de mensajes, etc.).

La última versión de SOAP define cualquier dato en términos de XML, la W3C recomienda que se defina un conjunto de estándares de terminologías para referenciar la información presentada en XML. La versión 1.2 también ha mejorado la gestión de errores. Esta versión clarifica el envío de errores para el mejoramiento y posteriormente definir el elemento con error.

El estilo de codificación SOAP también a sido mejorado, clarificado y algunos cambios han sido realizados para trabajar con HTTP. La disponibilidad de abrir o descargar archivos por mensajes SOAP, incrementa en gran medida la versatilidad.

Cuando las aplicaciones transmiten archivos sobre Internet, los datos son enviados en algún tipo de formato binario, como por ejemplo GIF, JPEG, JPG. Los mensajes de SOAP con documentos adjuntos usan MIME (Multipurpose Internet Mail Extensiones) para describir varios tipos de medios, tales como audio y video, tanto, que lo puede utilizar al ser procesados por aplicaciones. Por lo contrario cuando se envían archivos XML se utilizan mensajes SOAP.

Microsoft creo su propia especificación, la cual fue DIME (Direct Internet Message Encapsulation) para dar flexibilidad a las funciones del MIME.

3.3.3 UDDI

UDDI (Universal Description, Discovery, and Integration) es una especificación para crear registros distribuidos de servicios en red. Es una iniciativa industrial de importantes fabricantes y desarrolladores de software, incluyendo Microsoft, IBM, SUN, Ariba y HP. Un registro UDDI contiene información sobre empresas, sobre los servicios ofrecidos por estas empresas e información técnica sobre esos servicios. El modelo de datos y la interfaz de programación (API) de UDDI, basados en XML y SOAP, proporcionan métodos para publicar y localizar toda clase de servicios[10].

UDDI representa actualmente la capa de descubrimiento dentro del protocolo stack de los Web Services. UDDI esta compuesta de dos partes [9]:

1. Es una técnica de especificación para la construir de un directorio distribuido de negocios o servicios. Los datos almacenados en UDDI tienen un formato XML.
2. El UBR (UDDI Business Registry) es una implementación completamente operacional de la especificación UDDI. Lanzada en mayo del 2001 por Microsoft e IBM.

Un registro UDDI se puede comparar con un buscador en Internet, puesto que el buscador contiene información indexada y categorizada sobre las páginas Web. A diferencia del buscador, que únicamente devuelve un URL de una página Web, un registro UDDI necesita ofrecer no sólo la localización de los servicios, sino también información sobre el servicio, cómo funciona, qué parámetros utilizar, qué valores devuelve, etc.

El descubrimiento puede ser categorizado en directo e indirecto:

- ✓ El descubrimiento directo es el proceso de obtener datos desde un registro mantenido por el proveedor de servicio.
- ✓ En el descubrimiento indirecto, la organización obtiene datos por un intermediario o tercer usuario del registro.

Un registro es una entrada en UDDI, el cual contiene información a cerca de servicios que proporciona el Web Services, por lo general el UBR es accesible públicamente, por lo que existe un descontento por las grandes compañías, a las cuales les interesa demasiado mantener sus servicios disponibles solo para sus socios y personas involucradas con la organización, esto llevo a las empresas a crear registros privados, dando integridad y seguridad a cada uno de los servicios que provee. Con estas restricciones, las organizaciones son las que deciden quien tiene derecho a observar y manipular la información de los servicios que se encuentran en los registros de UDDI, obviamente privados.

Existen cuatro variantes de los registros privados:

- ✓ *e-marketplace UDDI:* Es un registro que aloja una industria y lista de negocios que se encuentra dentro de un consorcio de empresas.
- ✓ *portal UDDI:* Es un registro que reside en un firewall de la compañía y contiene información acerca de un simple Web Services de la organización. La compañía publica información acerca de sus servicios en el registro para que otras compañías puedan verlo.
- ✓ *partner catalog UDDI:* Es un registro que reside atrás del firewall de la compañía y enlista lo que el Web Services ofrece por una organización o sus socios.

- ✓ *internal enterprise application integration UDDI:* Se encuentra detrás del firewall de la compañía, pero solo puede ser accesado por usuarios internos.

Los registros UDDI contienen la información general y técnica acerca de los negocios y sus Web Services. Los vendedores regularmente comparan la estructura UBR con el directorio telefónico. El UBR principalmente soporta descubrimiento indirecto, puesto que aloja cuatro intermediarios. De cualquier manera UDDI también puede soportar descubrimiento directo en registros privados. Esto es por que los registros privados utilizados son implementados por organizaciones especificas y solo describen servicios ofrecidos por estas organizaciones.

UDDI provee un repositorio centralizado para publicar información técnica acerca de Web Services. Los repositorios siguen el modelo de replicación single-master, esto quiere decir que existe equipo de computo distribuido por la web, el cual se coordina con uno denominado master, este ultimo, tiene el objetivo de propagar los cambios a los demás repositorios.

El operador del repositorio UDDI genera un identificador único para cada tipo de datos cuando la información es publicada por primera vez en el repositorio. UDDI define una estructura compleja que contiene la información necesaria para descubrir los servicios e interactuar con ellos.

3.3.3.1 Limitaciones

Algunas limitaciones que se encuentran en el funcionamiento y desarrollo de este estándar se muestran a continuación de acuerdo a [5]:

- ✓ Una limitación significativa de UDDI es su inmadurez, ya que aun no ha sido sometida por alguna organización oficial que valide sus estándares.
- ✓ Aun que las compañías pueden buscar y encontrar la infamación deseada, UDDI no es capaz de mostrar la fecha en la que los datos fueron actualizados.

3.3.4 WSDL

WSDL (Web Services Description Language) es un formato XML que describe los servicios de red como un conjunto de puntos finales que procesan mensajes contenedores de información orientada tanto a documentos como a procedimientos. Las operaciones y los mensajes se describen de forma abstracta y después se enlazan a un protocolo de red y a un formato de mensaje concreto para definir un punto final de red. Los puntos finales concretos relacionados se combinan en puntos finales abstractos (servicios). WSDL es extensible, lo que permite la descripción de puntos finales de red y sus mensajes, independientemente de los formatos de los mensajes o protocolos de red utilizados para comunicarse. [10]

Una vez que se ha desarrollado un servicio Web, se publica su descripción y se construye una liga o apuntador en un deposito UDDI para que los usuarios potenciales lo puedan utilizar. Cuando alguien piensa en utilizar este Web Services, solicitan el archivo WSDL para conocer la ubicación del servicio, llamado de funciones y como acceder al Web Services. Luego utilizan la información en el archivo WSDL para construir una petición SOAP y enviarla hacia el proveedor de servicio.

Una de las ideas centrales detrás de los Web Services es que las aplicaciones futuras estarán conformadas de una colección de servicios habilitados en la red. Mientras haya dos servicios equivalentes que se publiciten a la red de una forma estándar y neutra, en teoría una aplicación podría seleccionar uno de ellos en base a criterios establecidos de antemano como precio o rendimiento. Además,

algunos servicios podrían permitir que fueran copiados entre maquinas, permitiendo así que una aplicación que corra en una maquina (o cluster de maquinas) mejore en rendimiento al copiar servicios útiles a unidades de discos locales.

Se podría hacer una analogía de esta situación con la del mercado de trabajo. Las compañías de contrataciones proveen un servicio de aparejamiento entre trabajadores y empleados, utilizando los resúmenes personales o currículo vitae y las descripciones de los trabajos ofertados para facilitar el proceso de búsqueda. Si se encuentra una buena opción, las partes interesadas intentan negociar condiciones aceptables para ambas. Si se logra un acuerdo, el trabajo se traslada a la nueva empresa o usa las ventajas del Internet y trabajar a distancia.

3.3.5 Funcionamiento general de un Web Services

En la Figura 3.3, se muestra de una manera más explicativa como es que funciona un Web Services, al momento que un cliente realiza una petición a un servicio, y se observa que pasos son los que generalmente se realizan para recibir, procesar y responder a dicha petición.

Otra cuestión es que se muestra en que momento intervienen cada uno de los estándares descritos en este capitulo.

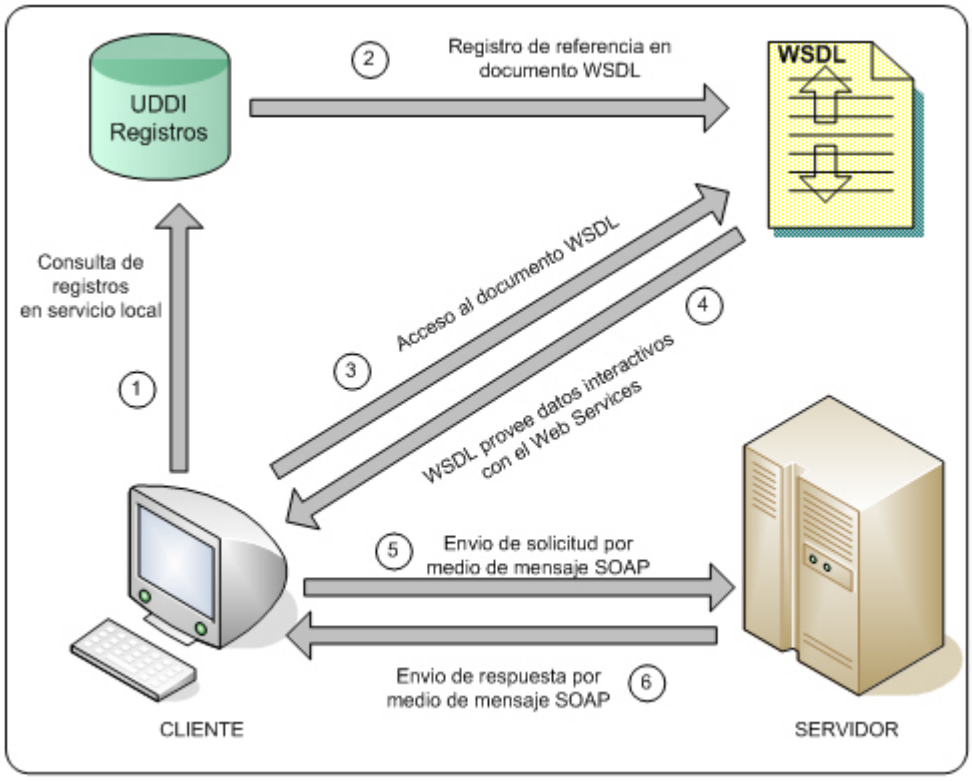


Figura 3.3 Funcionamiento Web Services con los estándares.

3.4 Resumen

Como se menciona al inicio de este capítulo, Internet esta evolucionando rápidamente de los sitios web actuales, que simplemente proporcionan paginas de interfaz de usuario a través de exploradores, a una generación de sitios web que brinden servicios basados en la interacción de programas para brindar servicios, esta característica amplía las posibilidades de explotación de bases de datos, servicios y aplicaciones a través de la web.

Es por ello, que en este capitulo se introducen y describen conceptos acerca de Web Services y se muestra la diferencia que existe entre los Web Services y los Web Servers. Y una parte importante, es el conocimiento de los estándares de diseño y operación que existen para poner en practica para implementar un Web Services, estos estándares son: UDDI, XML, WSDL y SOAP.

Por ultimo, en la sección 3.3.5 de este capítulo, se muestra un ejemplo general del funcionamiento de un Web Services utilizando los estándares antes mencionados, en el cual, se puede observar claramente cual es el flujo de la información, sirviéndonos esto como base para la implementación de un Web Services en el capítulo 6.

Capítulo 4

gSOAP

4.1 Introducción

Pensando en crear un Web Services eficiente y que además se pueda desarrollar con herramientas que no representen ningún costo monetario, una buena idea es utilizar gSOAP, puesto que aparte de cumplir con las características anteriores, es independiente de la plataforma (puede trabajar sobre Windows, Unix, Linux, Pocket PC, Mac OS X, TRU 64, VxWorks, etc.).

La herramienta gSOAP [6] cuenta con bibliotecas, dichas bibliotecas contienen primitivas para el desarrollo de aplicaciones basándose en lenguaje C y C++, las cuales acceden a Web Services utilizando el envío de mensajes con formato SOAP, con lo cual se le evita al desarrollador las complejidades inherentes a los protocolos de comunicación, haciendo de esta herramienta un apoyo muy útil. Además, proporciona una herramienta de compilación que provee automáticamente los tipos de datos nativos de C y C++, así como los que el desarrollador define a tipos SOAP semánticamente equivalentes, lo cual produce una interoperabilidad total de su producto.

Las bibliotecas de gSOAP han sido desarrolladas en Florida State University, E.E.U.U, esta herramienta es de dominio público y puede descargase desde su sitio en internet [6].

En las siguientes secciones de este capítulo se hará referencia a algunos ejemplos de cómo ejecutar comandos, estos ejemplos son aplicables para la plataforma Linux.

4.2 Benchmark de gSOAP y Axis

Los siguientes resultados (ver Tabla 4.1) fueron obtenidos por The Institute of Computer Science [7] de la republica Checa, al realizar comparaciones de implementaciones de Web Services para diferentes sistemas operativos y procesadores, cabe mencionar que las aplicaciones utilizadas tanto de cliente,

como servidor eran equivalentes para cada una de las implementaciones, con el fin de obtener los resultados más exactos.

Al ser gSOAP un preprocesador, es capaz de generar código con un solo propósito, mientras que Axis puede crear llamadas dinámicas muy generales para SOAP, por lo que los resultados muestran que gSOAP es mucho más rápido. El sistema operativo que se utilizo para las pruebas fue Linux 2.4.23 y el CPU fue el Pentium 4 a 2.5GHz

Llamadas por segundo	Aplicación Cliente	Aplicación Servidor
1540	gSOAP 2.5, gcc 2.95.4 -O2	gSOAP 2.5, gcc 2.95.4 -O2
375	Axis, Java IBM 1.4.1SR1	gSOAP 2.5, gcc 2.95.4
370	gSOAP 2.5, gcc 2.95.4	Axis, Java SUN 1.5.0-b -server -XX
360	gSOAP 2.5, gcc 2.95.4	Axis, Java IBM 1.4.1SR1
352	Axis, Java SUN 1.5.0-b -server -XX	gSOAP 2.5, gcc 2.95.4
310	Axis, Java SUN 1.5.0-b -client	gSOAP 2.5, gcc 2.95.4
198	Axis, Java SUN 1.5.0-b -server -XX	Axis, Java SUN 1.5.0-b -server -XX
187	Axis, Java IBM 1.4.1SR1	Axis, Java IBM 1.4.1SR1

Tabla 4.1 Web Services SOAP probado con gSOAP y Axis.

Existen otros resultados (ver Tabla 4.2) realizados utilizando un cliente y servidor implementados en gSOAP, ejecutándolos sobre diferentes plataformas.

Llamadas por segundo	S.O	Versión y compilador	CPU
2340	Linux 2.4.21 64-bit	gSOAP 2.5, gcc 3.2.2 -O2 64-bit	2x AMD Opteron 244 1.8GHz
2130	Linux 2.4.21 64-bit	gSOAP 2.5, gcc 2.95.4 -O2 32-bit	2x AMD Opteron 244 1.8GHz
1765	Linux 2.4.23	gSOAP 2.5, gcc 2.95.4 -O2	2x Xeon P4 3.06GHz
1750	Linux 2.4.23	gSOAP 2.5, gcc 3.3.1 -O2	2x Xeon P4 3.06GHz
1600	Linux 2.4.23	gSOAP 2.5, gcc 2.95.4	2x Xeon P4 3.06GHz
1530	Linux 2.4.23	gSOAP 2.5, gcc 3.3.1	2x Xeon P4 3.06GHz
1590	Linux 2.4.19 IA-64	gSOAP 2.5, Intel ecc 7.0 -O2	2x Itanium2 1GHz
1514	Linux 2.4.19 IA-64	gSOAP 2.5, gcc 2.96 -O2	2x Itanium2 1GHz
1540	Linux 2.4.23	gSOAP 2.5, gcc 2.95.4 -O2	1x Pentium4 2.5GHz
1430	Linux 2.4.23	gSOAP 2.5, gcc 2.95.4	1x Pentium4 2.5GHz
703	AIX 5.2	gSOAP 2.5, gcc 2.9-aix51 -O2	2x Power4+ 1.2GHz
530	SunOS 5.8	gSOAP 2.5, gcc 2.8.1 -O2	2x UltraSPARC-III 750MHz

Tabla 4.2 Web Services gSOAP en diferentes plataformas.
Las graficas están presentadas en una escala de 8 veces por debajo de lo real.

4.3 Previo a iniciar el desarrollo de un servicio

- ✓ Primeramente el software tiene que ser descargado (seleccionando la plataforma que se utilizará). Se recomienda utilizar la versión 2.6, por ser la más actual.
- ✓ Desempacarlo en el directorio en donde se desee trabajar. De los archivos obtenidos, con los que se trabajaran directamente son los siguientes:

    - **soapcpp2**: El cual es un archivo ejecutable, encargado de compilar los archivos.h que el desarrollador proporciona, con el fin de crear algunos archivos básicos (.xml, .nsmap, .xsd) para el buen funcionamiento del servicio.
    - **stdsoap2.cpp y/o .c**: El cual es un archivo librería, este archivo será requerido al utilizar la librería **gcc** o **g++,** al momento de crear las aplicaciones; Cliente y servidor.
    - **Wsdl2h:** El cual es un archivo ejecutable, es utilizado cuando el desarrollador cuenta con el archivo. wsdl y desea generar el archivo .h para sus aplicaciones.

4.4 Desarrollo de un servicio

El procedimiento para desarrollar los clientes utilizando C++ parte de un archivo de descripción del servicio externo especificado en WSDL y genera automáticamente archivos con el código fuente y headers que compilados producen el cliente para comunicación con el servicio remoto.

Un ejemplo de lo anterior se logra siguiendo la sintaxis:

- ✓ **wsdl2h -o outfile.h infile.wsdl**
- ✓ **wsdl2h -o Amazon.h http://soap.amazon.com/schemas/AmazonWebServices.wsdl**

Donde *infile.wsdl* puede ser un archivo del desarrollador (primer ejemplo) o puede estar alojado en un sitio Web(segundo ejemplo). Los archivos *outfile.h y Amazon.h* son el resultado de esta ejecución, los cuales contienen las definiciones de las cabeceras que serán utilizadas en los archivos *.c* o *.cpp* del cliente y servidor del Web Services.

Como consecuencia de la existencia de diferentes estándares de descripción de servicios, en algunos casos la herramienta gSOAP no es capaz de producir automáticamente los clientes, requiriendo esfuerzo adicional de programación. Para lo cual se deben proporcionar los documentos que se utilizaran para generar tanto el cliente, como el servidor.

Para lograr lo anterior, es necesario ejecutar el compilador de gSOAP (soapcpp2) sobre el archivo cabecera *.h*, con el fin de generar el código fuente que servirán para la implementación del servicio. El compilador genera los stubs y skeletons (y RPC stubs) para la aplicación Cliente[6].

Un ejemplo de la sintaxis para la generación de los archivos anteriormente mencionados seria la siguiente:

- ✓ **soapcpp2 wsmac.h**

Donde el archivo *wsmac.h* fue creado según las necesidades del desarrollador.

gSOAP en tiempo de ejecución provee una capa de transporte sobre HTTP, el cual trabaja sobre el protocolo TCP/IP, así como también, soporte para SSL y DIME.

Ahora que ya se cuentan con los archivos *.h, .wsdl*, *.xml*, etc., se procede a utilizar la herramienta gcc o g++ para compilar y ligar estos archivos con los .cpp o .c que el desarrollador previamente tuvo que haber realizado, en los cuales se programó lo suficiente para que la aplicación cliente pueda realizar la comunicación con el servidor y viceversa.

Nota: es muy importante que dentro de los archivos *.cpp o .c* (tanto en el cliente, como en el servidor) se hayan agregado adecuadamente los name spaces (estas líneas se encuentran en el archivo *.nsmap*, el cual fue generado por la herramienta), de lo contrario no empataran adecuadamente y en tiempo de ejecución se obtendrá un error al intentar ser enlazados.

La sintaxis más común para generar los archivos ejecutables a partir de los fuentes es la siguiente:

- ✓ **g++ -o wsmac wsmac.cpp soapC.cpp soapServer.cpp stdsoap2.cpp**

Donde *wsmac* es el nombre del archivo ejecutable, *wsmac.cpp* es el archivo que contiene las instrucciones necesarias para realizar el servicio, *soapC.cpp* y *soapServer.cpp* son los archivos que fueron generados por la herramienta y que contienen los formatos de los mensajes SOAP que serán utilizados para intercambio de información y por ultimo el archivo *stdsoap2.cpp* el cual es proporcionado por gSOAP al ser desempacado. Cabe mencionar, que el ejemplo

anterior es aplicado para generar la aplicación servidor, para generar el cliente la sintaxis seria la siguiente:

- ✓ **g++ -o wsclient wsclient.cpp soapC.cpp soapClient.cpp stdsoap2.cpp**

En donde se puede observar que la variante es del archivo *soapClient.cpp* y obviamente el nombre que se desea dar a la aplicación cliente.

4.5 Ejemplo de implementación de cliente

Este ejemplo es proporcionado por The Florida State University, y el cual parte del entendido de que existe un archivo *.wsdl* contenido en la Web, alguna información que contiene este archivo se describe a continuación:

Endpoint URL:	http://services.xmethods.net:80/soap
SOAP action:	"" (2 quotes)
Remote method namespace:	urn:xmethods-delayed-quotes
Remote method name:	GetQuote
Input parameter:	symbol of type xsd:string
Output parameter:	Result of type xsd:float

Ahora se realiza la conversión del documento *.wsdl* a el archivo cabecera *.h*, utilizando el comando *wsdl2h*, utilizando la siguiente línea:

- ✓ ***wsdl2h -o quote.h http://services.xmethods.net/soap/urn:xmethods-delayed-quotes.wsdl***

El archivo *quote.h* que ha sido generado contiene lo siguiente:

```
//gsoap ns1 service name: quote
//gsoap ns1 service type: net_DOT_xmethods_DOT_services_DOT_stockquote_DOT_StockQuotePortType
//gsoap ns1 service port: http://66.28.98.121:9090/soap
//gsoap ns1 service namespace: urn:xmethods-delayed-quotes
//gsoap ns1 service documentation: Definitions generated by the gSOAP WSDL parser 1.0
// Service net.xmethods.services.stockquote.StockQuoteService :
net.xmethods.services.stockquote.StockQuote web service
//gsoap ns1 service method-style: getQuote rpc
//gsoap ns1 service method-encoding: getQuote http://schemas.xmlsoap.org/soap/encoding/
//gsoap ns1 service method-action: getQuote urn:xmethods-delayed-quotes#getQuote
int ns1__getQuote(char *symbol, float &Result);
```

Enseguida se procede a generar los archivos *.xml, .xsd, .nsmap,*etc. Con la siguiente sintaxis:

- ✓ **soapcpp2 -c quote.h**

El siguiente paso es compilar el archivo *cliente.cpp*, el cual contiene:

```
#include "soapH.h" // obtain the generated stub
#include "Quote.nsmap" // obtain the generated XML namespace mapping table for the Quote service
main()
{
  struct soap *soap = soap_new();
  float quote;
  if (soap_call_ns1__getQuote(soap, NULL, NULL, "IBM", quote) == SOAP_OK)
    printf("Current IBM Stock Quote = %g\n", quote);
  else // an error occurred
    soap_print_fault(soap, stderr); // display the SOAP fault on the stderr stream
}
```

- ✓ **g++ -o wsclient cliente.cpp soapC.cpp soapClient.cpp stdsoap2.cpp**

4.6 Ejemplo de implementación de Servicio

Si contamos con el archivo *calc.h* desarrollado por el programador, el cual contiene:

```
//gsoap ns service name: calculator
//gsoap ns service style: rpc
//gsoap ns service encoding: encoded
//gsoap ns service port: http://mydomain/path/calculator.cgi
//gsoap ns service namespace: urn:calculator
int ns__add(double a, double b, double &result);
int ns__sub(double a, double b, double &result);
int ns__sqrt(double a, double &result);
```

Se generan los archivos skeletons (*.xml,xsd,etc*):

- ✓ **soapcpp2 calc.h**

Teniendo el archivo *calc.cpp*, el cual es el servicio y contiene:

```
#include "soapH.h"
#include "calculator.nsmap"
#include < math.h >
main()
{
  soap_serve(soap_new()); // call the incoming remote method request dispatcher
}
// Implementation of the "add" remote method:
int ns__add(struct soap *soap, double a, double b, double &result)
{
  result = a + b;
  return SOAP_OK;
}

// Implementation of the "sub" remote method:
int ns__sub(struct soap *soap, double a, double b, double &result)
{
  result = a - b;
  return SOAP_OK;
}

// Implementation of the "sqrt" remote method:
int ns__sqrt(struct soap *soap, double a, double &result);
{
  if (a > = 0)
  {
    result = sqrt(a);
    return SOAP_OK;
  }
  else
  {
    return soap_sender_fault(soap, "Square root of negative value", "I can only compute the square root of a
non-negative value");
  }
}
```

Se genera el archivo ejecutable para ser utilizado como el servicio:

- ✓ **g++ -o wscalc calc.cpp soapC.cpp soapServer.cpp stdsoap2.cpp**

4.7 Resumen

En este capítulo se describe la herramienta gSOAP, las características con las que cuenta, los fines para los cuales fue creado y sobre que lenguajes de programación se pueden realizar las implementaciones de Web Services.

Con el objeto de realizar las evaluaciones de las dos técnicas de encriptación (elementos y datos) que utilizaremos, se pensó en desarrollar un Web Services con herramientas que no representen ningún costo monetario y agilice el servicio, resultando una buena idea utilizar gSOAP, puesto que aparte de cumplir con las características anteriores, es independiente de la plataforma (puede trabajar sobre Windows, Unix, Linux, Pocket PC, Mac OS X, TRU 64, VxWorks, etc).

En base a lo anterior, y en especifico en las secciones 4.4, 4.5 y 4.6 es como se realizará la implementación del Web Services que se utilizará en este trabajo de investigación, aplicado en el capítulo 6.

Capítulo 5

Modelo Propuesto

5.1 Introducción

En los capítulos anteriores se han descrito los estándares bajo los cuales los Web Services debe de ser implementado, así como la importancia que tiene la seguridad y cifrado de la información para este tipo de desarrollos, debido a la exposición que tienen por naturaleza, al acceso prácticamente de cualquier persona por utilizar la infraestructura de la Internet. Y por ultimo en el capitulo anterior se mostró como es que se puede implementar un Web Services utilizando la herramienta gSOAP de creación basada en C/C++.

Partiendo de lo anterior e intentando aplicar todos estos conceptos, herramientas y algoritmos, para enriquecer este trabajo de investigación, es necesario contar con un modelo, el cual servirá como base para realizar el desarrollo del servicio sobre el cual se evaluaran las técnicas de encriptación, con la intención específica de conducir un estudio que permitirá determinar la forma en la que se comportan las diferentes técnicas de encriptación.

5.2 Modelo

A continuación se muestra como es que se puede plantear un modelo de servicio (ver Figura 5.1), que cubra con todas las especificaciones para ser considerado como Web Services y que se puedan implementar los algoritmos criptográficos para realizar las evaluaciones adecuadas.

Nota : Debido a que todos los elementos que son utilizados en este modelo ya fueron cubiertos en los capítulos anteriores, sus descripciones no aparecen en esta sección. Pero en esta sección se describe la simbología utilizada en al Figura 5.1 y Figura 5.2.

Representa las aplicaciones (Cliente y Servidor) que intervienen en el intercambio de mensajes SOAP.

Representa el SOAP engine (utilizando las bibliotecas SOAP).

Representa los archivos que fueron serán generados en la implementación y en tiempo de ejecución.

Representa la dirección tanto del flujo de información, como de peticiones y respuesta a un servicio en específico.

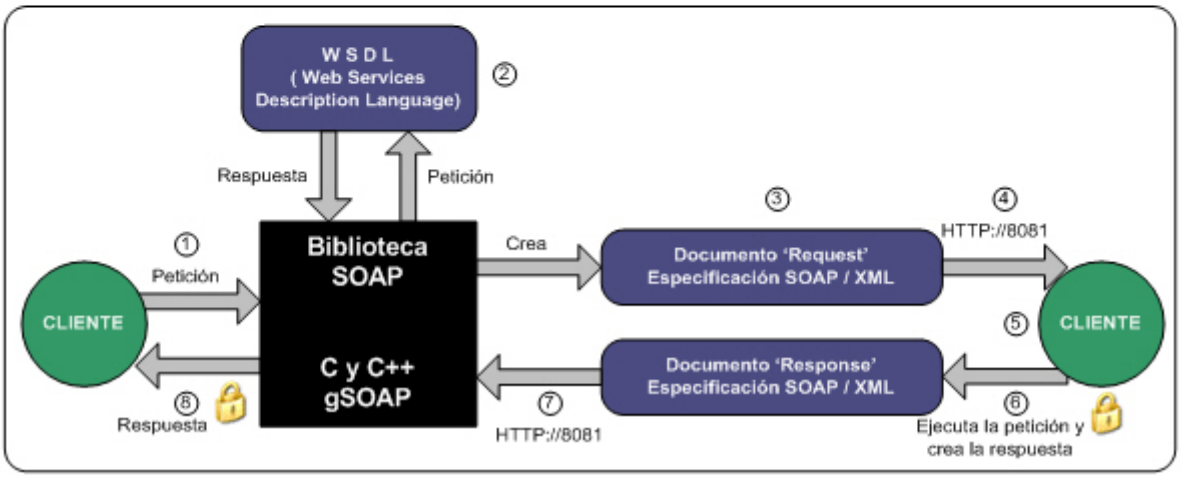


Figura 5.1 Flujo de información con encriptación de elementos en XML.

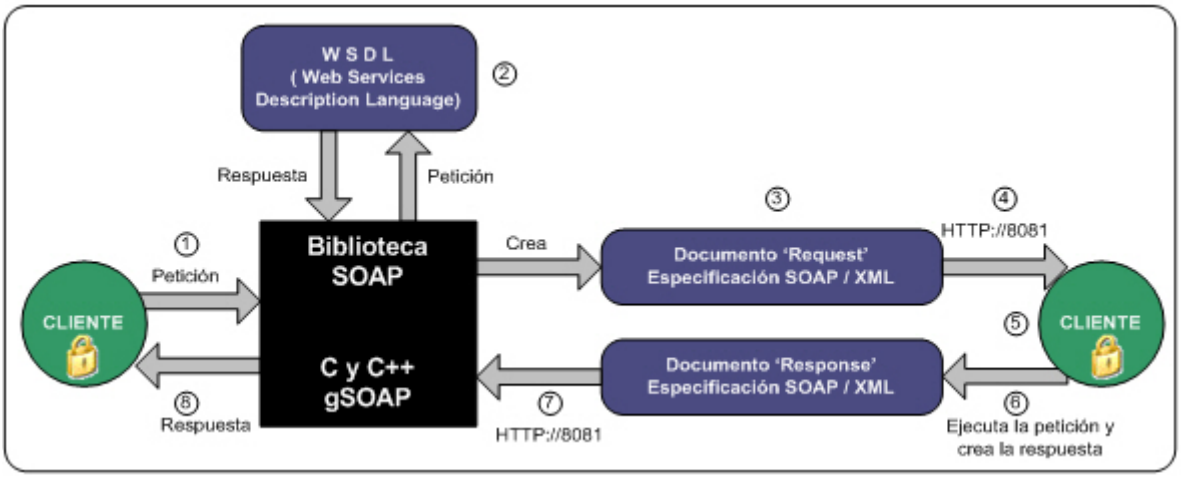


Figura 5.2 Flujo de información con encriptación de datos con algoritmos.

Los paso que se siguen para lograr tener acceso al servicio y recibir una respuesta correcta a la petición realizada son los siguientes:

1. La aplicación “Cliente” toma los datos a ser enviados y encripta los datos, posteriormente genera una petición del servicio en especifico.
2. Esta petición es tomada por las bibliotecas de SOAP (implementadas con gSOAP) y busca en sus plantillas WSDL, que la petición cumpla con la sintaxis y características adecuadas para el servicio solicitado.
3. Al recibir la información anterior, las bibliotecas SOAP generan el documento que contendrá la petición en un formato SOAP.
4. Esta información es enviada vía HTTP hacia la dirección valida de la aplicación “Servidor” y el puerto el cual dará el servicio (en este ejemplo al 8081).
5. La aplicación “Servidor” (el cual previamente tuvo que ser ejecutado, para estar a la espera de peticiones de servicio) recibe el documento con el formato SOAP, el cual es capaz de entender, debido a que al ser generado el “Servidor” gSOAP le agrego funciones para poder convertir los datos a un formato entendible para él. Se valida la entrada, se realiza el desencriptado de los datos, se ejecutan las instrucciones programadas para brindar el servicio. Al terminar la ejecución, los datos son encriptados según la forma selectiva que se utilizo por parte del cliente.
6. Se provee la respuesta para el servicio solicitado y se envía a la dirección que genero la petición, vía HTTP, la respuesta creando un nuevo documento que contiene un formato SOAP.
7. SOAP engine efectúa la tarea de validar la entrada con sus plantillas WSDL y UDDI, y entrega el documento al “Cliente”.
8. Por ultimo, el “Cliente” recibe el documento en formato SOAP y lo convierte a datos entendibles por la aplicación, y para ser desencriptan los datos que fueron encriptados por el “Servidor”.

5.3 Resumen

Es importante entender como es que se realiza el flujo de los datos que se utilizan al ejecutar el servicio que se pretende implementar, ya que en base a ello es que se pueden visualizar de manera más sencilla los puntos clave en los cuales el desarrollador debe de colocar los algoritmos criptográficos, para asegurar la información y no comprometerla para que alguna persona no autorizada pueda tener acceso a estos datos, que dependiendo del servicio es que se puede ponderar el valor de los mismos.

Con lo visto en este capitulo, se podrá entender de una manera clara como es que se logra la implementación que se describirá en el siguiente capitulo.

Capítulo 6

Implementación

6.1 Introducción

Teniendo como antecedente los capítulos anteriores, tanto en conceptos como en el modelo a seguir, en este apartado se presenta la implementación de los algoritmos criptográficos y un ejemplo de un Web Services, para lograr obtener los resultados que sustenten el tema propuesto para este trabajo de investigación.

Se muestra primeramente el escenario a seguir, en el cual se describe el tipo de servicio que se brinda y como es que se realizan las operaciones para solicitar y recibir la información de forma integra. Como punto importante, se presentan los pseudo códigos de los algoritmos criptográficos y aplicaciones Cliente y Servidor, con los cuales se pretende lograr la implementación del Web Services.

Como parte adicional se muestran los archivos que son generados por la herramienta gSOAP al momento de crear el servicio.

6.2 Escenario

Se pretende generar un Web Services, el cual pueda ser capaz de brindar un servicio el cual sea capaz de brindar la integridad, confidencialidad y consistencia, lo anterior tomando en consideración que para la implementación y pruebas se realizaran en el equipo de computo, y con las características de software descritos en 1.4.

El servicio a simular es un validador de datos para realizar una transacción monetaria, en la cual se involucran datos del usuario como lo son:

- ✓ Nombre
- ✓ Dirección
- ✓ Numero de Tarjeta de crédito
- ✓ Tipo de compra
- ✓ Monto

Este servicio recibirá los datos del usuario, los validara y para efectos de este trabajo devolverá los mismos datos después de haber agregado una información de aceptación.

6.3 Implementación del Web Services

Para la implementación del servicio se utiliza la herramienta gSOAP, a continuación se presentan los archivos desarrollados, así como los que se generan después con esta herramienta de generación.

Paso 1: Generación de los documentos partiendo del archivo cabecera *wsmac.h*, el cual contiene la descripción de la sintaxis que se va a utilizar para realizar el intercambio de información. La parte importante del contenido de este archivo es el siguiente:

```
//gsoap ns service name: ws-sample
//gsoap ns service location: http:/127.0.0.1
//gsoap ns schema namespace: http://www.itesm.mx/
int ns__descrip(char *desenvio,char * &desrecivo);
int ns__nombre(char *nenvio,char * &nrecivo);
int ns__direccion(char *denvio,char * &drecivo);
int ns__tarjeta(char *tenvio,char * &trecivo);
int ns__tipo_compra(char *cenvio,char * &crecivo);
```

La instrucción que se utiliza para la generación de los archivos plantilla que se utilizarán es la siguiente:

- ✓ **soapcpp2 wsmac.h**

Después de ejecutar este comando se generan los siguientes archivos:

Ns.xsd
SoapC.cpp
SoapClient.cpp
SoapH.h

SoapServer.cpp
SoapServerlib.cpp
Soapws-sampleObject.h
Soapws-sampleProxy.h
Ws-sample.nombre_del_dato_a_solicitar.req.xml
Ws-sample.nombre_del_dato_a_responder.res.xml
Ws-sample.wsdl

Paso 2: Se realiza la programación de los archivos *wsserver.cpp* y *wsclient.cpp*, los cuales contienen principalmente el código que realizara el servicio, por parte del servidor y del cliente respectivamente. De manera general se presenta el contenido de estos archivos (No el código).

Archivo *wsserver.cpp:*

- ✓ Las librerías necesarias para la utilización de funciones propias de C++.
- ✓ La librería *SoapH.h*, la cual contiene las cabeceras que utiliza en general gSOAP para el intercambio de información utilizando SOAP.
- ✓ La implementación del algoritmo criptográfico, tanto la función de encriptación, como desencriptación, para ser utilizada antes de enviar los datos y recibirlos respectivamente.
- ✓ La función principal, la cual es la encargada de iniciar la conexión al servicio, así como de recibir las peticiones de la aplicación Cliente y la llamada a las funciones definidas en el archivo cabecera (*wsmac.h*).
- ✓ Las estructuras de las funciones que gSOAP creó, con las instrucciones que se desean realizar al momento de ser llamadas por el programa principal.
- ✓ La declaración del *namespace* , la cual debe ser exactamente igual tanto en el programa Cliente como Servidor (estas líneas de código se deben copiar del archivo *ws-sample.nsmap*. De no ser idénticos en este punto las aplicaciones no podrán realizar la conexión entre ellos.

Archivo *wsclient.cpp:*

- ✓ Las librerías necesarias para la utilización de funciones propias de C++.
- ✓ La librería *SoapH.h*, la cual contiene las cabeceras que utiliza en general gSOAP para el intercambio de información utilizando SOAP.

- ✓ La función que se utilizara para la lectura de los datos, que posteriormente serán manipulados por las aplicaciones (se utiliza esta función debido a que para realizar las evaluaciones se utilizarán cadenas de caracteres muy largas).
- ✓ La implementación del algoritmo criptográfico, tanto la función de encriptación, como des-encriptación, para ser utilizada antes de enviar los datos y recibirlos respectivamente.
- ✓ La función principal, la cual es la encargada de iniciar la conexión al servicio, así como de generar las peticiones de la aplicación Servidor y la llamada a las funciones definidas en el archivo cabecera (*wsmac.h*), en este punto es donde se configura el puerto en el cual buscará el servicio.
- ✓ Las estructuras de las funciones que gSOAP creó, con las instrucciones que se desean realizar al momento de ser llamadas por el programa principal.
- ✓ La declaración del *namespace* , la cual debe ser exactamente igual tanto en el programa Cliente como Servidor (estas líneas de código se deben copiar del archivo *ws-sample.nsmap*. De no ser idénticos en este punto las aplicaciones no podrán realizar la conexión entre ellos.

Paso 3: Se generan los archivos ejecutables con las siguientes instrucciones.

Para generar la aplicación Servidor:

- ✓ **g++ -Wno-deprecated -o wsserver wsserver.cpp soapC.cpp soapServer.cpp stdsoap2.cpp -DLEANER -Os –lpthread**

Para generar la aplicación Cliente:

- ✓ **g++ -Wno-deprecated -o wsclient wsclient.cpp soapC.cpp soapClient.cpp stdsoap2.cpp -DLEANER -Os**

Nota: Los pasos anteriores se realizan dos veces, debido a que para esta investigación y obtención de los resultados, como ya se mencionó en 2.4, se utilizaran dos algoritmos criptográficos, y debido a esto, los archivos *wsserver.cpp* y *wsclient.cpp* deben ser modificados para la implementación de cada uno de ellos.

Paso 4: Por ultimo para realizar la ejecución de estas aplicaciones y levantar el servicio se deben ejecutar las siguientes instrucciones en este orden:

- ✓ **./wsserver localhost 8081 o ./wsserver 127.0.0.1 8081**, con esta instrucción se habilitará la aplicación Servidor y estará en

espera de las peticiones por parte del Cliente vía HTTP por el puerto 8081.

- ✓ **./wsclient,** al ejecutar esta instrucción se comenzara con la petición del servicio.

Nota: Cada instrucción debe de ser ejecutada en diferentes consolas de Linux.

6.4 Resumen

En este capítulo se define el escenario a seguir, en el cual se describe el tipo de servicio que se brinda y como es que se realizan las operaciones para solicitar y recibir la información de forma integra. Como punto importante, se presentan los pseudo códigos de los algoritmos criptográficos y aplicaciones Cliente y Servidor, con los cuales se pretende lograr la implementación del Web Services.

Posteriormente, se muestra paso a paso como es que fue desarrollado el Web Services utilizando la herramienta gSOAP (Ver capítulo 4), así como la forma en que se ejecutan cada una de las aplicaciones para realizar el servicio.

Capítulo 7

Pruebas y Resultados

7.1 Introducción

Dada la naturaleza de este trabajo, se requiere realizar implementaciones y mediciones, las cuales basadas en parámetros reales, brindaran y fundamentarán la evaluación de los métodos de encriptación aplicados a un rango de valores predeterminados. En este capitulo se muestran dichos parámetros y los resultados que se obtuvieron al llevar a la practica los conceptos mostrados en los capítulos anteriores.

7.2 Parámetros utilizados

Como se mencionó en el punto 2.4, los algoritmos criptográficos que a utilizar son: AES (Advanced Encryption Standard), At-Bash (intercambio de caracteres) y ElGamal (utilizando el teorema de curvas elípticas).

Para evaluar cada uno de estos algoritmos implementados dentro de las aplicaciones Cliente y Servidor del Web Services creado (ver capitulo 6), se utilizan cadenas de caracteres de diferentes tamaños, cabe mencionar que el tamaño total que se muestra en Bytes es la suma de cadenas pequeñas que simularon varias transacciones electrónicas, las cuales son leídas desde archivos de texto(la alimentación se realiza con valores aleatorios, para evitar que las implementaciones generaran patrones y las mediciones erróneas). Las dimensiones de dichas cadenas se muestran tabla 7.1:

Tamaño en Bytes	Caracteres contenidos
50 KB	50,986
100KB	101,972
500KB	511,940
1MB	1,023,880
1.5MB	1,535,820

Tabla 7.1 Tamaños de cadenas de caracteres.

Nota: Las cadenas anteriores utilizan un tipo de dato caracter, del estándar ANSI.

El algoritmo At-Bash utilizado, es implementado con un valor de desfase de caracter de tres posiciones. Un ejemplo simple de cadena que es cifrada seria la siguiente:

- ✓ Cadena original: **abcdefghi.**
- ✓ Cadena encriptada: **defghijkl.**

El algoritmo de ElGamal utilizado como se mencionó en el punto 2.4.1, requiere de varios parámetros para generar su clave criptográfica y, realizar el cifrado y descifrado. A continuación se muestra el listado de variables involucradas y como es que se seleccionaron.

a) Generación de claves:

1. Tomar un valor x y un grupo finito Zn {x, x^2 ,x^3, ...x^n}. Sea G un valor calculado que se encuentra dentro de Zn.
   Sea n el cardinal de G.
   En este trabajo se definió **x = 3** y **n = 10.**
   Resultando el grupo **Zn = {3,9,27,81,243,729,2187,6561,19683,59049}.**

2. Se selecciona un número aleatorio **a**, entre 1 y n-1, y calcular **3^a** (buscar el valor en el grupo **Zn**).
3. La clave pública es el par (**3,3^a**). La clave privada es **a**.

b) Cifrado:

5. Se selecciona el caracter a cifrar y en se utiliza como valor **m** para esta formula.
6. Se selecciona aleatoriamente **k**, entre 1 y n-1.
7. Calcular **G = 3^k y f = m(3^a)^k** , donde **f** es el valor encriptado.
8. El criptograma c es el par (**G,f**).

c) Descifrado:

1. Calcular **aG** y luego **–aG** (su opuesto en la curva).
2. Dando como resultado **M=(-aG)+F,** donde **M** es el valor original.

Ejemplo:

m = 122 (valor de z en ANSI), x = 3, n = 5, a = 2 y k = 4.
Zn = {3,9,27,81,243,729,2187,6561,19683,59049}

Cifrado: **G = 3 ^ k = 9**
f = m (3^a)^k = 122(9)^4 = 800442.
a^G = 81^2= 6561, G^-a = 1.5241579027587e^{-4}

Descifrado: **M = (G^-a) *f =** (1.5241579027587e^{-4}) * 8004 **= 122.**

7.3 Resultados

Las comparaciones de tiempos se basaron en el cifrado de valores aleatorio (los que se definieron como sensitivos) y todos valores (que están descritos en la estructura de datos) que son utilizados para el intercambio de mensajes SOAP.

A continuación se muestran los resultados obtenidos (Tabla 7.2, 7.3 y 7.4) para cada uno de las técnicas de encriptación, separadas por los tres tipos de algoritmos criptográficos que se utilizaron.

	Algoritmo At-Bash		
	Tamaño	Tiempo	
		Hora:Min:Seg	Hora:Min:Seg
1	50k	0:00:15	0:00:19
2	100k	0:01:11	0:01:52
3	500k	0:03:36	0:06:37
4	1m	0:06:49	0:19:35
5	1.5m	0:10:31	1:15:10
	Valores	**Sensitivos**	**Todos**

Tabla 7.2 Tiempos utilizando At-Bash.

	Algoritmo ElGamal		
	Tamaño	Tiempo	
		Hora:Min:Seg	Hora:Min:Seg
1	50k	0:00:43	0:01:35
2	100k	0:02:18	0:03:53
3	500k	0:05:02	0:07:55
4	1m	0:20:10	1:03:22
5	1.5m	0:40:02	2:47:02
	Valores	**Sensitivos**	**Todos**

Tabla 7.3 Tiempos utilizando ElGamal.

	Algoritmo AES-128		
	Tamaño	Tiempo	
		Hora:Min:Seg	Hora:Min:Seg
1	50k	0:01:52	0:04:02
2	100k	0:04:36	0:09:48
3	500k	0:12:49	0:30:06
4	1m	0:39:36	1:40:48
5	1.5m	1:14:10	4:10:16
	Valores	**Sensitivos**	**Todos**

Tabla 7.4 Tiempos utilizando AES-128.

Como resultado y para tener un mejor enfoque del comportamiento que se presentó con el tiempo de ejecución de Web Services, se muestran las siguientes graficas al utilizar diferentes tamaños de archivos, para el algoritmo At-Bash ver Figura 7.1, para ElGamal ver figura 7.2 y para AES-128 ver Figura 7.3.

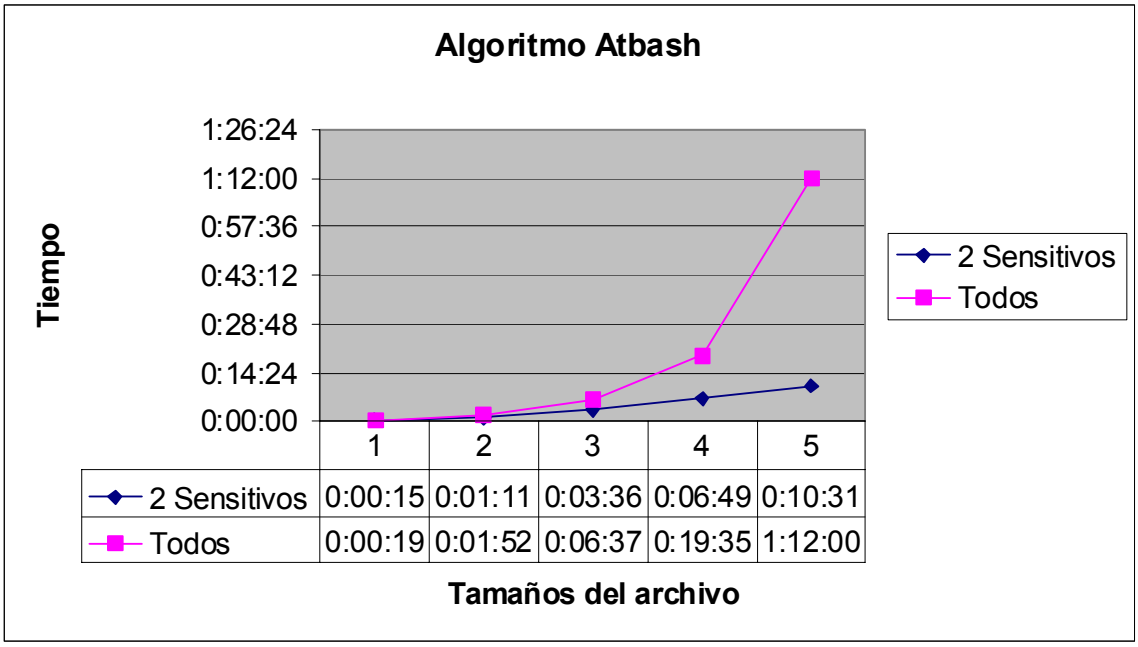


Figura 7.1 Tiempos utilizando At-Bash con dos y todos los valores de la estructura de datos.

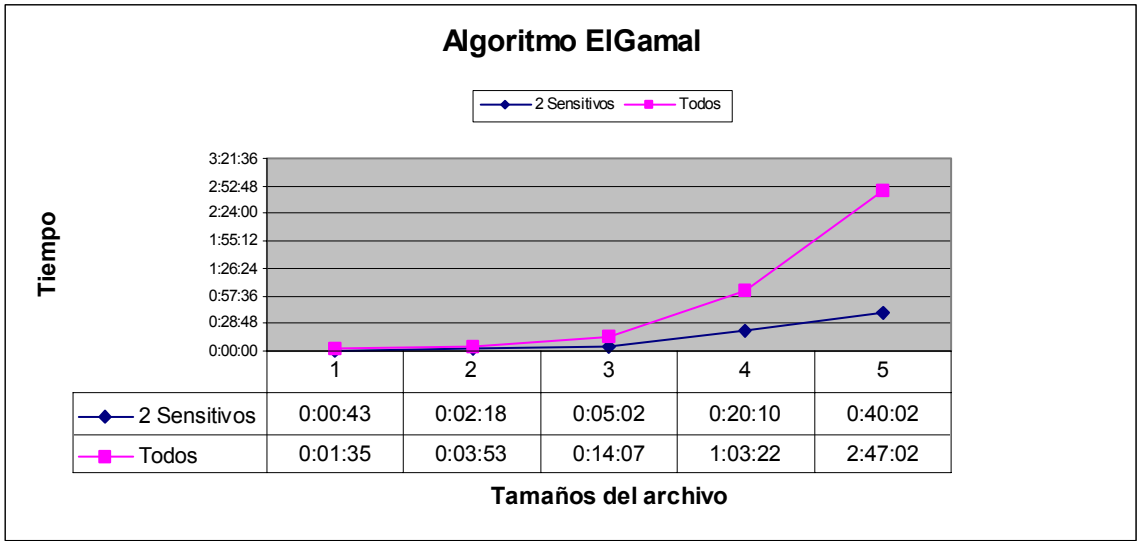


Figura 7.2 Tiempos utilizando ElGamal con dos y todos los valores de la estructura de datos.

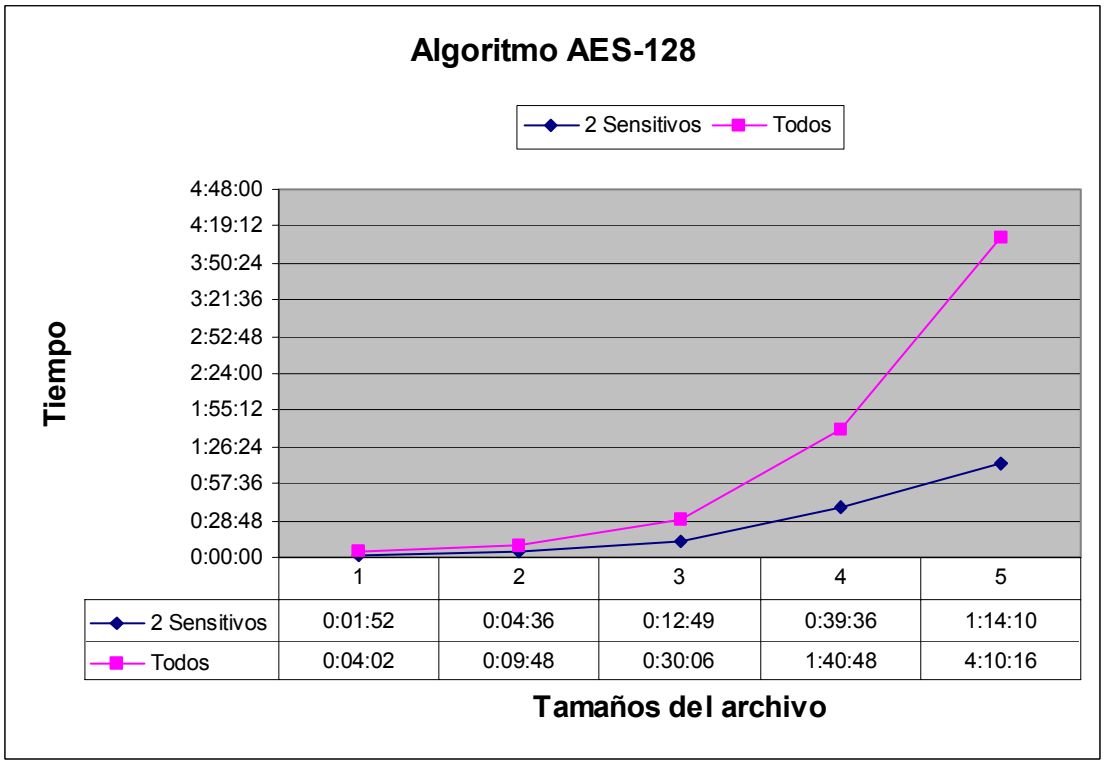


Figura 7.3 Tiempos utilizando AES-128 con dos y todos los valores de la estructura de datos.

Después de observar el comportamiento de los tres algoritmos criptográficos, se presentan a continuación los tiempos registrados en la ejecución al implementar el cifrado de elementos dentro del documento XML, utilizando librerías especiales para ser utilizadas en C++, la encriptación se realiza utilizando el algoritmo AES-128.

A continuación se muestran en la tabla 7.5 los resultados obtenidos para cada uno de las técnicas de encriptación, aplicada a la encriptación de elementos de un documento XML.

	XML Encryption AES-128		
	Tamaño	Tiempo	
		Hora:Min:Seg	Hora:Min:Seg
1	50k	0:11:31	0:19:44
2	100k	0:21:01	0:34:34
3	500k	0:37:09	1:04:13
4	1m	0:56:53	3:12:49
5	1.5m	1:55:12	4:33:36
	Valores	**Sensitivos**	**Todos**

Tabla 7.5 Tiempos utilizando XML Encryption AES-128.

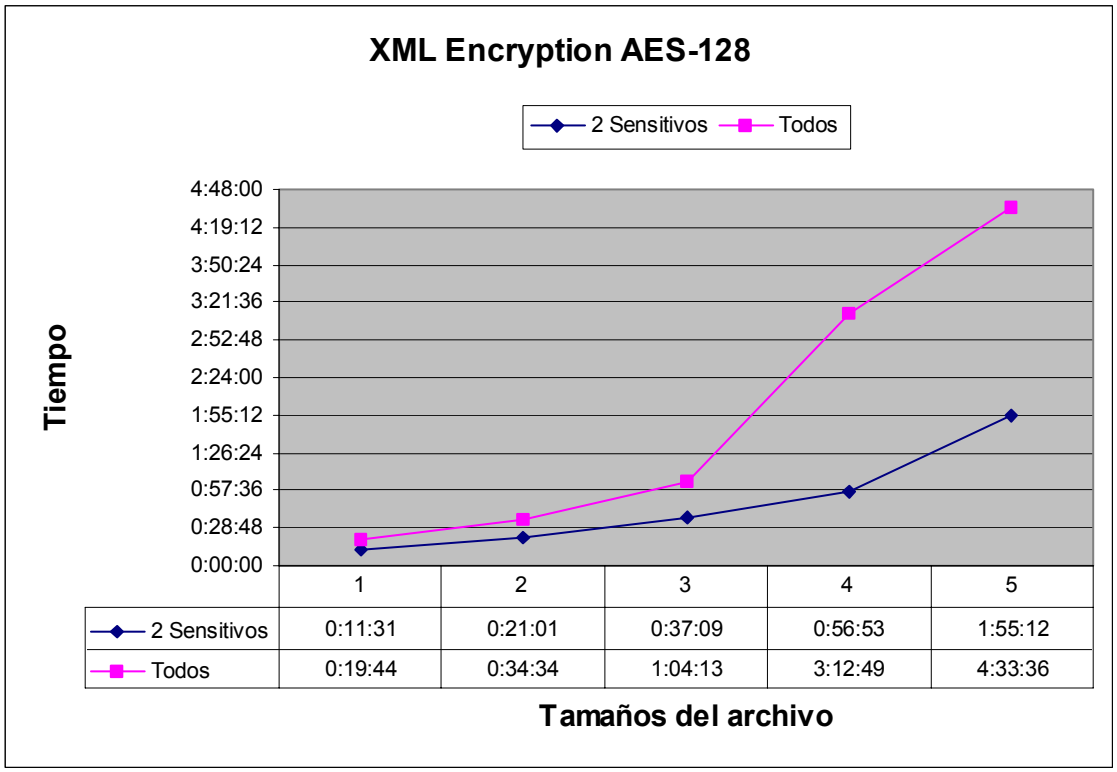
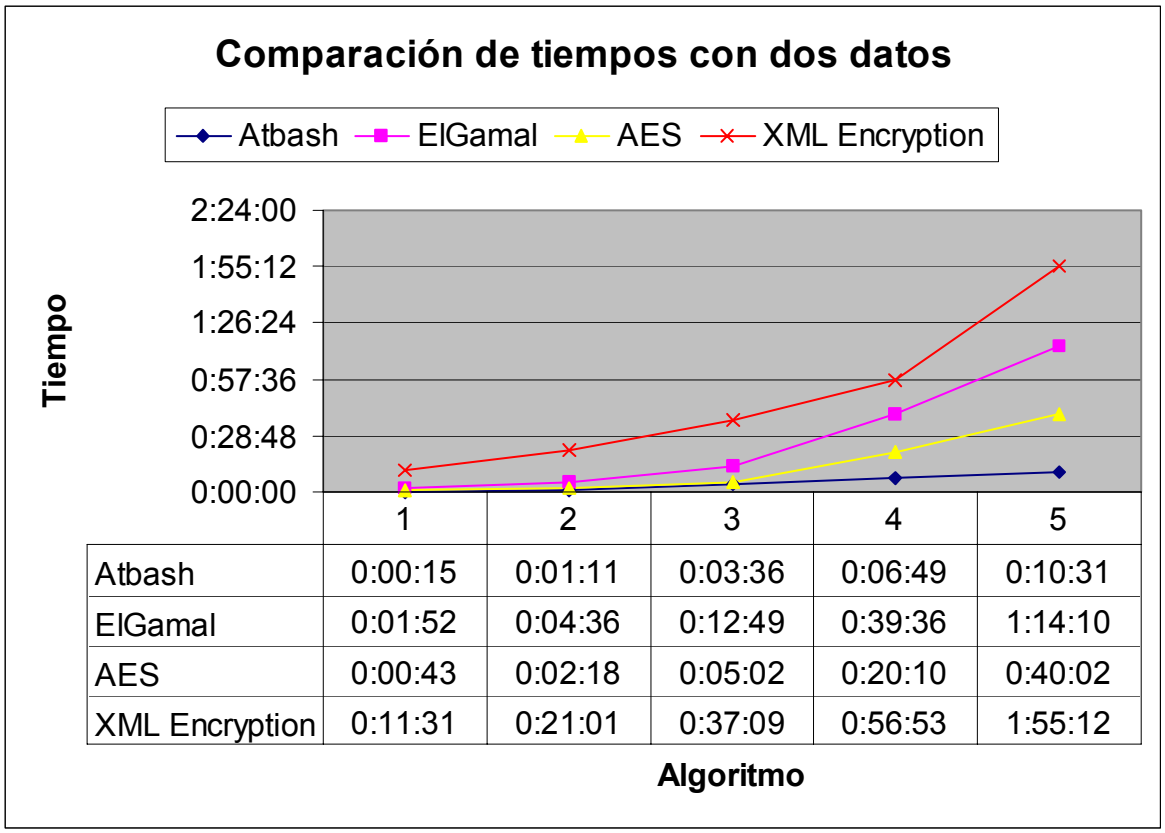


Figura 7.4 Tiempos utilizando XML Encryption AES-128 con dos y todos los valores de la estructura de datos.

A continuación se muestran los comparativos de tiempos, resultado de las ejecuciones de los tres algoritmos criptográficos aplicados a los datos y la implementación del XML Encryption sobre elementos.

La figura 7.5 muestran los resultados al encriptar dos datos sensitivos de la estructura de datos definida para el ejemplo de servicio que se simuló.



	1	2	3	4	5
Atbash	0:00:15	0:01:11	0:03:36	0:06:49	0:10:31
ElGamal	0:01:52	0:04:36	0:12:49	0:39:36	1:14:10
AES	0:00:43	0:02:18	0:05:02	0:20:10	0:40:02
XML Encryption	0:11:31	0:21:01	0:37:09	0:56:53	1:55:12

Figura 7.5 Algoritmos VS XML Encryption AES-128 con dos de los valores de la estructura de datos.

La figura 7.6 muestra el resultado al encriptar todos los datos de la estructura definida para este ejemplo de servicio.

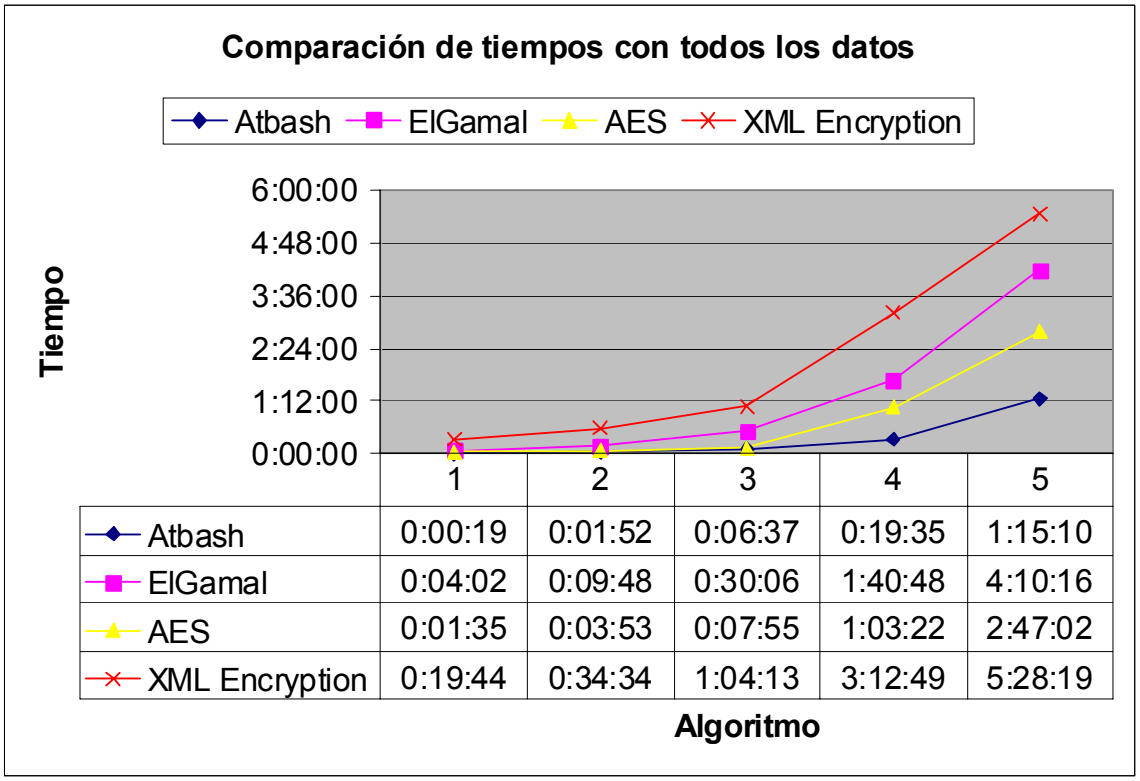


Figura 7.6 Algoritmos VS XML Encryption AES-128 con todos los valores de la estructura de datos.

Es importante mencionar que después de aplicar el algoritmo de ElGamal, el tamaño de la cadena encriptada varia al original, debido a que al encriptar cada uno de los caracteres se convierten en tipo de dato *long int*, ya que se hacen los cálculos de cifrado y descifrado con este tipo de datos, los valores resultantes son los siguientes Tabla 7.5:

Antes Tamaño en Bytes	**Después** Tamaño en Bytes	Caracteres contenidos
50 KB	200 KB	50,986
100KB	400KB	101,972
500KB	2MB	511,940
1MB	4MB	1,023,880
1.5MB	6MB	1,535,820

Tabla 7.5 Tamaños de cadenas de caracteres después de ser encriptada con ElGamal.

Capítulo 8

Conclusiones

8.1 Introducción

En este capitulo se desarrollan los razonamientos, resultado de los datos arrojados por la implementación de los tipos de encriptación sobre documentos XML en Web Services que realicé. Así mismo, presento las conclusiones de la propuesta de este trabajo de investigación (ver Capitulo 1), y fundamento las mismas en lo presentado en los Capítulos 5, 6 y 7, los cuales, respaldan los resultados obtenidos.

Lo anterior me lleva a finalizar este trabajo con las siguientes aportaciones, previstas en Capitulo 1.

8.2 Aportaciones

- ✓ Observando el comportamiento que se presentó, utilizando la encriptación de dos valores, clasificados como sensitivos para las implementaciones de los algoritmos criptográficos (Figuras 7.1,7.2 y 7.3), puedo concluir que:

  Presenta un crecimiento casi lineal ascendente, este crecimiento esta determinado por el tamaño de la cadena que se esta encriptando.

- ✓ Utilizando el mismo criterio de análisis (Figuras 7.1,7.2 y 7.3), pero ahora para todos los valores (5 con los que cuenta la estructura de datos), puedo concluir que:

  Presenta un crecimiento lineal partiendo de información con dimensión de 50KB, pero al ir en incremento el tamaño de la información se observa que el crecimiento se comporta exponencial, contrario a lo que se podría pensar, después de haber observado la comparación anterior.

- ✓ Utilizando la encriptación de los elementos del documento XML, con dos valores sensitivos y otra implementación encriptando todos los valores de la estructura de datos que se utilizó (Figura 7.4), observé:

  Al igual que en los algoritmos criptográficos que se aplicaron para la encriptación de datos, el crecimiento se comportó de forma similar al la encriptación de elementos del documento XML, con la diferencia que sus tiempos de ejecución fueron mayores a los que se observaron para los 3 algoritmos al cifrar datos.

La conclusión a la que llego, después de las observaciones anteriores, es la siguiente:

La hipótesis de esta tesis se ve sustentada en los tiempos registrados después de realizar la implementación y ejecución de la encriptación de datos y de elementos de documentos XML, ya que los tiempos de ejecución observados y mostrados en las Figuras 7.5 y 7.6, me permite afirmar que la encriptación de datos con cualquiera de estos tres algoritmos criptográficos consume un menor tiempo de ejecución que la encriptación de elementos recomendada por el estándar de la W3C, utilizando la infraestructura de los Web Services.

Capítulo 9

Trabajos futuros

9.1 Introducción

Desde que nace el concepto de Web Services, han surgido algunas preguntas sobre su seguridad y derivado a la poca madurez que se presenta en la mayoría de las especificaciones, se había puesto poca atención a la seguridad, pero esto, esta cambiando hoy en día, ya que poco a poco se ha ido convirtiendo de una simple pregunta en un problema a resolver. Y ya que el interés sobre la seguridad en los Web Services va en incremento, hay aspectos que sin duda alguna son añadidos a ella, como por ejemplo, el rendimiento, rapidez y manejo eficaz de los datos al momento de realizar los intercambios de mensajes mediante el estándar SOAP.

9.2 Recomendaciones

Las recomendaciones que surgen a partir de esta investigación son:

- ✓ Algunos experimentos por falta de tiempo y de capacidad de equipo de computo no fueron posibles realizar, como por ejemplo:

    1. Utilizar algoritmos criptográficos con llaves más grandes.
    2. Utilizar una mayor cantidad de algoritmos criptográficos.
    3. Utilizar un ambiente de pruebas con varios equipos de cómputo, simulando una red real de trabajo.

- ✓ Realizar esta misma implementación de técnicas de encriptación y algoritmos utilizando otro tipo de tecnologías, como Java.

- ✓ Esta misma comparación y aplicación de conceptos puede ser utilizadas entre un Web Services y dispositivos móviles.

Capítulo 10

Bibliografía

10.1 Referencias bibliografías

[1] Emin Gün Sirer, Ke Wang *: An access control language for web services* June 2002, ACM Press New York, NY, USA, Pages: 23 - 30 Series-Proceeding-Section-Article .

[2] Engin Kirda: *Web engineering device independent web services,* July 2001, ACM Press Washington, DC, USA, Pages: 795 - 796 Series-Proceeding-Article.

[3] http://www.ibw.com.ni/~gangeles/jerk/seguridad.html visitado: 18/Nov/2003, 18:00 horas.

[4] Mark O'Neill with Phillip Hallam-Baker, Seán Mac Cann, Mike Shema, Ed Simon, Paul A. Watters, and Andrew White, *Web Services Security,* January 2003, McGraw-Hill/Osborne , Berkeley, California, USA. Pages: 89- 99.

[5] H. M. Deitel, P. J. Deitel, B. DuWaldt, L. K. Trees, *Web Services: A technical Introduction.* 2003. Deitel Developer Series, Ed. Prentice Hall, Upper Saddle River, New Jersey.

[6] gSOAP: Herramienta generadora de código para crear Web Services basados en SOAP y aplicaciones cliente utilizando C++ y C. Florida State University. http://www.cs.fsu.edu/~engelen/soap.html
visitado: 28/Abril/2004, 13:00 horas.

[7] The Institute of Computer Science en Masaryk University, Republica Checa http://www.ics.muni.cz/~makub/java/speed.html#ws visitado: 2/Mayo/2004

[8] Kriptópolis: http://www.kriptopolis.com/more.php?id=176_0_1_0_M visitado: 29/Abril/2004, 10:00 horas.

[9] Ethan Cerami: *Web Services Essentials*, O'REILLY, Gravenstain Highway North, Sebastopol, CA 95472, First Edition 2002.

[10] Dietrich Ayala, Christopher Browne, Vivek Chopra, Dr. Poornachandra, Kapil Apshankar, Tim McAllister: *Professional Open Source Web Services*, Wrox Press Ltd 2002, UK, United States.

[11] http://www.reduy.com/computacion/ms-com-electronico/technet-7.htm visitado: 30/Junio/2004, 11:30 horas.

[12] Bruce Scheneier: *Criptografía aplicada*, John Wiley & Sons, 1996, 2ª Edición. Capitulo 7.

[13] http://www.virusprot.com/Nt121221.html, visitado: 17/Julio/2004, 16:00 horas.

[14] Ignacio Alfredo Badillo, René a. Cumplido Parra, *Reporte técnico no. Ccc-04-006,* Coordinación de ciencias computacionales: INAOE, 28 de Junio del 2004.