PRACTICA EMPRESARIAL: IMPLEMENTACIÓN EN SOA (ARQUITECTURA ORIENTADA A SERVICIOS) DE LOS OBJETOS NECESARIOS PARA INTERCONECTAR LOS SISTEMAS TRANSACCIONALES DE GAZEL CON LA PLATAFORMA DE LA FIDUCIARIA BOGOTÁ A TRAVÉS DE ATH.

VERA ASTRID CABRERA ESPITIA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2010

PRÁCTICA EMPRESARIAL: IMPLEMENTACIÓN EN SOA (ARQUITECTURA ORIENTADA A SERVICIOS) DE LOS OBJETOS NECESARIOS PARA INTERCONECTAR LOS SISTEMAS TRANSACCIONALES DE GAZEL CON LA PLATAFORMA DE LA FIDUCIARIA BOGOTÁ A TRAVÉS DE ATH.

VERA ASTRID CABRERA ESPITIA

Proyecto de Grado presentado para optar por el título de Ingeniero de Sistemas

Director
Héctor Niño Quiñonez
Ingeniero de sistemas

Codirector
Carlos Armando Galeano
Ingeniero de Sistemas

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2010

DEDICATORIA

A Dios por no soltar mi mano nunca. Por que su palabra se ha cumplido y ha llegado el momento. Porque para siempre es su misericordia.

A mi Amada Madre, quien es mi mejor ejemplo y mi fortaleza. Porque se lo merece , porque sin ella no hubiese sido posible.

A mi hermano , por mi infinito amor hacia él. Por su valentía, inteligencia y los anhelos de su corazón.

A mis queridas amigas , Natalie, Margie, Yoleida y churcos , por ser mi gran compañía y apoyo. Por las sonrisas y lágrimas compartidas, porque este triunfo es la culminación del gran proyecto que un día iniciamos y el que nunca dejamos atrás por más fuerte que fuese la tormenta.

A mi querido amigo Alvaro a quien admiro profundamente, por su inteligencia y disciplina. Por los detalles que se ganaron un lugar en mi corazón.

Homenaje especial a Guillermo Antonio Araque Montes por creer en mi . Por ser quien era. Por su valentía y su forma de ver la vida , por las sonrisas y la confianza. Porque para siempre estará en mi corazón y su ejemplo en mis días de batalla.

verita

AGRADECIMIENTOS

En primer lugar a Dios todo poderoso por darme la oportunidad de seguir adelante sin dejar de batallar a pesar de las pruebas y tribulaciones durante estos años de aprendizaje. Gracias a Él, por esta gran bendición.

A mi amada madre y adorado hermano por ser mi motor para vivir. Por su esfuerzo y sacrificio . pero sobre todas las cosas por no desfallecer en ningun momento.

Al Señor Miguel Angel Caro , quien me brindo la oportunidad de realizar este proyecto en su empresa Flexiweb LTDA. Agradecimientos por su apoyo, comprension y confianza. A cada uno de los ingenieros de esta gran empresa, quienes me acompañaron y propiciaron mi aprendizaje en todo momento.

Al Ingeniero Carlos Galeano, por ser guía en la elaboración de este proyecto. Quien fue de gran respaldo. Con su experiencia, fue de gran ejemplo a seguir, en el ambiente laboral al que me enfrentaba.

A mi querido amigo Joaquín quien oriento mi trabajo y fue pieza clave en su elaboración y culminación. Gracias a sus conocimientos, disciplina y por creer en mi.

Al profesor Hector Niño, quien me colaboro siendo el director de este proyecto y quien fue de gran apoyo para la construcción de este libro.

TABLA DE CONTENIDO

TABLA DE ILUSTRACIONES

RESUMEN

Título: IMPLEMENTACIÓN EN SOA (ARQUITECTURA ORIENTADA A SERVICIOS) DE LOS OBJETOS NECESARIOS PARA INTERCONECTAR LOS SISTEMAS TRANSACCIONALES DE GAZEL CON LA PLATAFORMA DE LA FIDUCIARIA BOGOTÁ A TRAVÉS DE ATH.*

Autor: Vera Astrid Cabrera Espitia **

Palabras Clave: Web Services, Protocolo, XML, Tecnología, Seguridad

Descripción:

Servicios Web es la tecnología que consiste en un conjunto de protocolos de mensajería y estándares abiertos de programación que exponen las funciones de negocio a través de Internet basándose en XML. Un servicio web individual es un componente discreto, software reutilizables que se tiene acceso mediante programación más Internet, a través de HTTP o SMTP a veces, para devolver una respuesta. La integración de aplicaciones se produce al permitir que los sistemas con diferentes software, puedan compartir o agregar información.

La web de los servicios de extensión de seguridad es un conjunto de características opcionales de SOAP. Se incluye la firma XML, cifrado XML, que se utiliza para cifrar los datos XML para proporcionar confidencialidad. Numerosas fichas de seguridad son compatibles con WS-Security, permitiendo que la identidad portátiles (SAML), el transporte de clave pública (X.509), información de gestión de derechos (XrML), y otros. XML Signature es la última y mejor tecnología para usted para garantizar la integridad y no repudio.

Esta práctica es orientada a la construcción de un servicio web, en el cual se implemente la seguridad solicitada por ATH, con el cual se generen extractos de los clientes de Gazel. Empresas Colombianas lideran el desarrollo del país.

***Trabajo de Grado. Modalidad: Práctica Empresarial.**

**** Facultad de ingenierías Físico-Mecánicas. Escuela de Ingeniería de sistemas e informática Director: Héctor Niño, ingeniero. Tutor: Carlos Galeano, Ingeniero.**

ABSTRACT

Title: IMPLEMENTATION IN SOA (ORIENTED ARCHITECTURE TO SERVICES) OF THE NECESSARY OBJECTS TO INTERCONNECT THE TRANSACTIONAL SYSTEMS DE GAZEL WITH THE PLATFORM OF THE FIDUCIARY BOGOTÁ TO INCLINATION DE ATH.*

Author: Vera Astrid Cabrera Espitia **

Keywords: Web Services, Protocol, XML, Technology, Security

Description:

Web services is the term for a technology that consists of a set of messaging protocols and programming open standards that expose business functions over the Internet, based on XML. An individual web service is a discrete component; reusable software is accessed programmatically over the Internet via HTTP or SMTP sometimes to return a response.

Application integration is produced by allowing different software systems to share or add information. The WS-Security is a set of optional features of SOAP. It includes XML signature, XML encryption, which is used to encrypt the XML data to provide confidentiality. Numerous safety cards are compatible with WS-Security, allowing portable identity (SAML), the transport public key (X.509), Information Rights Management (XrML), and others. XML Signature is the latest and best technology for you to ensure the integrity and non-repudiation.

Web Services have the potential to be the next great paradigm that will once and for all change computing. The great promise of WS will never be realized unless they are proven to be reliable, available, and have the appropriate level of security.

This practice is oriented at building a web service, which is implemented by ATH requested security, including security certificates, which are generated with extracts from the Gazel and customers traveling on the network without the inconvenience and optimally.

***Undergraduate Thesis: Modality: Business Practice.**

****Physics and Mechanics engineering Faculty. Systems Engieneering and Informatics School. Director: Héctor Niño, Engineer. Tutor: Carlos Galeano, Engineer**.

INTRODUCCION

El Gobierno Nacional, encabezado por la Presidencia de la República y los ministerios de Comunicaciones y educación, han comenzado una campaña para incentivar el uso de las Tecnologías de la información y la comunicación en empresas, centros educativos y en el gobierno. La idea es aprovechar las ventajas de la tecnología para generar desarrollo y competitividad, y por ende empleo y crecimiento económico[1].

Las organizaciones, actualmente dependen, en gran parte, de las tecnologías de la información y la comunicación (TIC) para alcanzar sus objetivos corporativos, por tanto que estas junto con los estándares basados en Web, reduzcan en gran medida los riesgos de integración y la complejidad, ofreciendo soluciones más específicas e independientes, flexibles y fáciles de combinar, modificar o ampliar. Empresarialmente se hace de gran importancia la construcción de relaciones duraderas y beneficiosas con los clientes como mecanismo competitivo frente al mercado que permita a las corporaciones mantenerse en la competitividad del mercado actual.

La Arquitectura Orientada a Servicios (SOA), es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos del negocio. Permite la creación de sistemas altamente escalables que reflejan el negocio de la organización, a su vez brinda una forma estándar de exposición e invocación de servicios, para nuestro caso un Servicio Web (SW), lo cual facilita la interacción entre diferentes sistemas propios o de terceros.

Entre las principales ventajas que ofrecen los SW se destaca el aumento de la productividad debido al incremento de fluidez en las relaciones entre proveedores, socios, empleados y clientes. Derivándose una mejor gestión, mantenimiento y actualización de la información. Se pueden observar conceptos muy ligados a los SW como son las tecnologías WSDL (*Web Services Description Language*) y UDDI (*Universal Description, Discovery, and Integration*), utilizados como estándares de descripción, publicación y registro de los Web Services (WS). Además de manejar el protocolo SOAP (*Simple Object Access Protocol*), estándar de comunicación para atender las peticiones y repuestas entre los diversos

[1] COLOMBIA SE CONECTA. Disponible en http://www,colombiaseconecta.gov.co/spip.php?article143 10 de abril de 2008. Citado [11 de agosto de 2009].

clientes y Servicios Web. El intercambio de datos se puede realizar entre distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma.

Para la empresa desarrolladora de Software, ***FLEXIWEB LTDA,*** siempre ha sido de gran importancia el estar a la vanguardia de dicho tipo de avances tecnológicos, y como gran reto está la creación e implementación del Servicio Web Seguro que permita a la FIDUCIARIA DEL BANCO DE BOGOTA, conectarse con la organización A Toda Hora (ATH), a través de su bus transaccional, lo cual que servirá como puente para intercambiar servicios con la empresa de gas vehicular GAZEL. Por parte del proveedor la puesta en marcha de este proyecto, es una gran oportunidad de abrirse a nuevos clientes, además de negocios en cualquier campo de aplicación en el que se requiera. Para ello requiere de una investigación profunda acerca del tema de los Web Services y las tecnologías necesarias para el desarrollo de la aplicación.

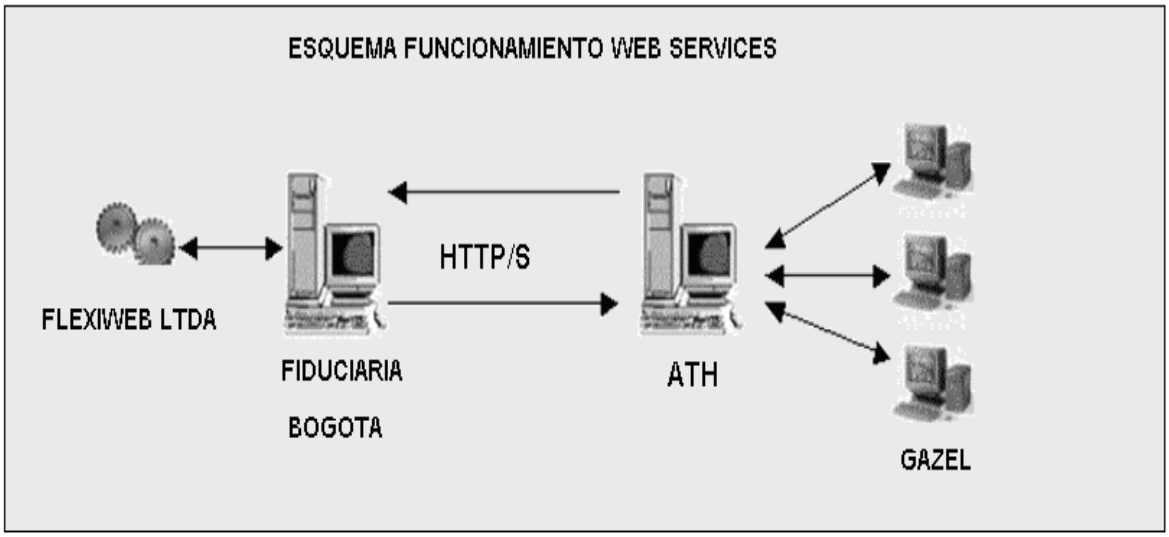


Ilustración 1 WSExtracto

1 FLEXIWEB LTDA

1.1 *DESCRIPCION DE LA EMPRESA*

Flexiweb Ltda, es una compañía colombiana de construcción de Software orientada a crear aplicaciones en JAVA, SAP (ABAP), utiliza como motor de base de datos Oracle y servidores Web como Tomcat y OC4J. En conjunto, son las herramientas utilizadas para el desarrollo de las aplicaciones, lo que certifica altos estándares de calidad.

Entre los aplicativos creados por esta compañía son:

- Aplicativo de Manejo de Afiliados a Fondos de Inversión (Carteras Colectivas)
- Aplicativo para el Manejo de Títulos Valores
- Aplicación FD para SAP, (Carteras Colectivas)
- Control de Fuerza de Ventas
- Asesoría para creación de Aplicaciones
- Creación de Web-Services para el desarrollo de sus aplicativos y para la exposición de servicios y productos que la compañía genere.

1.2 MISION

La empresa hace uso de las ultimas herramientas del mercado tendiente a mejorar el conocimiento de su grupo humano y generar aplicativos competitivos con los cuales los clientes de Flexiweb sean el horizonte en el sector.

Estar un paso adelante a las necesidades de nuestros clientes y así dar cumplimiento a sus requerimientos.

1.3 VISION

Ser a mediano plazo una de las empresas líderes en herramientas de punta orientadas al sector Fiduciario, así como tener el mejor equipo humano y técnico en la elaboración de software.

1.4 ORGANIGRAMA *DE LA COMPAÑÍA*

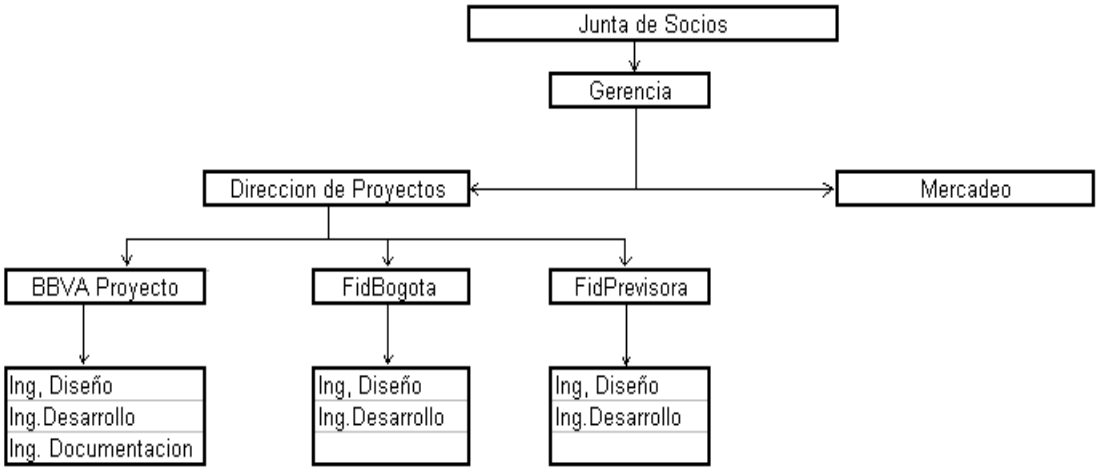


Ilustración 2 Organigrama Flexiweb LTDA

1.5 SERVICIOS

Dentro de los productos que se han desarrollado horizontalmente están:

- **FLEXIFON,** Sistema construido en JAVA sobre base de datos Oracle y WEB server OAS o TOMCAT, con capacidad para manejar hasta 999 fondos de inversión, interfaz con otros aplicativos.

- **MANEJADOR DE PORTAFOLIOS DE TÍTULOS VALORES**, con capacidad para calcular valores de mercado por medio de Precios tanto Limpios como Sucios, Tasas de Descuento, valores exponenciales y lineales de la inversión. Maneja múltiples portafolios con interfaz con otros aplicativos, y un simulador especial para proyecciones.

- **MERCAWEB,** Es el aplicativo para el seguimiento de la fuerza de ventas.

- **AUDITORWEB**, Es un aplicativo para mejorar y validar las cifras resultantes de la Auditoría Contable, con posibilidad de actualizar las cuentas y balances a partir de archivos planos, ayuda a la creación de ajustes y reclasificaciones con otras características.

- **SAP – FD**, En el momento está en desarrollo el sistema de afiliados a fondos de inversión y fondos pensiónales con ABAP para ser manejado en SAP.

2 JUSTIFICACIÓN

El valor concreto que aportan las tecnologías de la información es la eficacia. Esto se concluye de análisis acerca de las necesidades tecnológicas de las empresas intensivas en activos a lo largo de años de experiencia desarrollando soluciones de negocios para el sector empresarial. La tecnología se convierte en variable medible, es decir cuando permite que los procesos de gestión empresariales logren maximizar en términos porcentuales y cifras reales la rentabilidad de sus operaciones y la minimización de sus gastos operativos, administrativos y productivos.

Según el enfoque de Microsoft, la clave del éxito de las empresas radica en la capacidad de adaptación al cambio y en la optimización de procesos de negocios. La arquitectura orientada a servicios (SOA) es un enfoque que promueve la capacidad de las empresas para adaptarse al cambio. Con los Web Services se adquiere la aceleración en los procesos de comercio electrónico entre organizaciones, como respuesta a la integración de los componentes Software con los que cuentan, modificando sustancialmente la forma en que operan asumiendo riesgos tales como la seguridad.

Las preocupaciones del uso de WS radica en la potencial exposición de información corporativa, información de los clientes (quienes usan el servicio) y proveedores (quienes lo implementan) y la circulación de las transacciones fraudulentas. Actualmente se combinan tecnologías existentes, tales como el SSL (Secure Socket Layer), que permiten la transmisión encriptado, en conjunto con estándares específicos, tales como el SAML(Security Assertion Markup Language), que consiste en la autentificación y autorización basada en XML. Siendo de gran prioridad el incursionar en esta tecnología se hace necesario el indicar y confrontar diversas herramientas disponibles que proporcionen la opción de seleccionar una que se acomode a las necesidades de la empresa.

Para el caso de la Fiduciaria de Bogotá, se desea consolidar importantes acuerdos comerciales, basados en recursos, políticas y regulaciones financieras que están integradas en el Software que manejan y mecanismos de intercambio electrónico de información, con los que cuenta actualmente. El departamento de Sistemas de la FiduBogotá tiene como misión el ofrecer servicios fiables, de alta calidad y a un coste aceptable, para la optimización continua de sus procesos. A medida que las plataformas informáticas evolucionan, la dispersión de tecnologías, sistemas y

aplicaciones requieren de un servicio de atención al usuario de mejor calidad y con una mayor especialización.

Debido a la gran experiencia que posee FLEXIWEB LTDA, en el desarrollo de aplicaciones para el sistema de información de la FiduBogotá, se le ha encomendado un nuevo requerimiento, el cual consiste en desarrollar un Web Services (WS), que permita realizar transacciones a través de ATH con Gazel, y viceversa. La parte innovadora de este proyecto está en la utilización de WS, puesto que es la primera vez que se manejara este tipo de tecnología, junto con el manejo de la seguridad, importante para no causar conflictos con la conexión de ATH.

La creación de esta aplicación deberá permitir a futuro el intercambio financiero con GAZEL y la reutilización de este desarrollo para crear nuevas aplicaciones en posibles negocios de la Fiduciaria con otras entidades. Como estudiante se me ha encomendado la tarea de hacer parte de esta aplicación, puesto que es la oportunidad para adquirir conocimientos relacionados con los WS, el manejo de JAVA, JDeveloper 10g, y lo concerniente a la seguridad informática, que son parte del objetivo personal.

3 OBJETIVOS

3.1 *OBJETIVO GENERAL*

Implementar en la plataforma de la FIDUCIARIA BOGOTÁ, el Web Services (WS) requerido para conectarse al bus transaccional de ATH, que permitirá desarrollar servicios y procesos que se conectaran con la plataforma de servicios de GAZEL permitiendo realizar pre-aprobación de encargos, consulta de saldo y movimientos, depósitos y retiros .

3.2 *OBJETIVOS ESPECIFICOS*

- Realizar la investigación orientada al sistema de seguridad, para el Servicio Web, según requerimientos hechos por el usuario, en este caso ATH, basada en el Certificado X.509.

- Desarrollar e implantar modulo de seguridad, para el Servicio Web "**WSExtracto**", que generará los extractos bancarios para las transacciones de negocios exclusivas entre clientes de la Fiduciaria de Bogotá y ATH.

- Desarrollar e implantar modulo de verificación, el cual valide y evite errores que el usuario pueda cometer al intentar generar un extracto para un determinado cliente.

- Desempeñar actividades de monitoreo de ***HELP DESK***, dando solución a posibles fallas ya sean técnicas o errores encontrados en la información del Extracto.

- Elaborar la documentación técnica para "**WSExtracto**", junto con su manual de usuario.

- Propiciar la inducción a la vida laboral en un ambiente real, siendo parte del recurso humano con el que cuenta FLEXIWEB Ltda.

4 ESTADO DEL ARTE

Los Web Services surgieron de la necesidad de estandarizar la comunicación entre distintas plataformas (PC, Mainframe, Mac, entre otras) y lenguajes de programación (PHP, Java, C#, etc.). Anteriormente se realizaron intentos de crear estándares pero fracasaron o no tuvieron el suficiente éxito debido a que dependían de la implementación del vendedor DCOM, Microsoft, y CORBA-ORB(a pesar que CORBA de múltiples vendedores pueden operar entre sí, hay ciertas limitaciones para aplicaciones de niveles más altos en los cuales se necesite seguridad o administración de transacciones).

Adicionalmente, surge el problema al hacer uso de RPC (Remote Procedure Call) para realizar la comunicación entre diferentes nodos. Esto, además de presentar ciertos problemas de seguridad, tiene la desventaja de que su implementación en una ambiente como Internet, es casi imposible (muchos firewalls bloquean este tipo de mensajes, lo que hace prácticamente imposible a dos computadoras conectadas por Internet comunicarse).

Internet se fundamento en un conjunto de estándares de protocolos abiertos que se centran sobre el Protocolo de transporte hipertexto (*HyperText Transport Protocol o HTTP*) que se utiliza para compartir información entre máquinas. HTTP no reemplaza a TCP/IP, sino que es un protocolo de alto nivel que utiliza TCP/IP para la transmisión de bajo nivel.

Internet solidifico la dirección de los sistemas distribuidos, ya que mostró a las empresas que es posible aumentar la eficiencia de gran manera mediante una mejor utilización de las redes de computadoras. Sin embargo, se carecía de la capacidad de compartir servicios de Software que las empresas necesitaban para integrar completamente las aplicaciones de negocio de gran escala.

Surgió la siguiente evolución de los servicios de Software, conocida como Servicio Web (SW). Los SW comprenden una red de servicios, donde estos son las piezas de construcción de Software que se encuentran en una red, a partir de las cuales los programadores pueden crear sistemas distribuidos a gran escala.

Con la introducción de los SW se crearon nuevos estándares, WSDL, UDDI y el protocolo de arquitectura orientada a servicios. Los programadores utilizan el WSDL para publicar su servicio Web y que esté disponible a otros programadores a lo largo de la red, quienes utilizan la UDDI para localizarlos y el SOAP para llamar al servicio determinado.

Muchos sistemas distribuidos y SW a gran escala tienen algo en común. Están escritos con J2EE, debido a que este resuelve los problemas complejos que enfrenta un programador al desarrollar sistemas distribuidos a gran escala. En un sistema distribuido a gran escala típico se utilizan numerosos servicios Web, cada servicio se asocia a una capa en la arquitectura multicapa que se utiliza para compartir recursos a través de la infraestructura de la empresa.

Los Web Services surgieron para finalmente poder lograr la tan esperada comunicación entre diferentes plataformas. En la actualidad muchos sistemas Legacy están pasando a ser WS. Desde 1999 se comenzó a plantear un nuevo estándar, el cual terminaría utilizando XML, SOAP, WSDL y UDDI.

La Web como Servicio

Un servicio Web, es un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios Web para intercambiar datos en redes de ordenadores como Internet.

La interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS [2] y W3C [3] son los comités responsables de la arquitectura y reglamentación de los servicios Web. Para mejorar la interoperabilidad entre distintas implementaciones de SW se ha creado el organismo WS-I, encargado de desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares.

El protocolo Simple de acceso a objetos (SOAP) fue creado en 1997 por tres empresas de la industria: Microsoft, una pequeña empresa de software llamada Userland Software y DevelopMentor, empresa creada por Lucent, juntas se propusieron desarrollar un protocolo de comunicaciones que fuera fácil de utilizar

y suficientemente flexible para que pudiera aplicarse a los cambios en la industria y que proporciona la siempre esquiva interoperabilidad.
Poco después de comenzar el proyecto, centraron su atención en XML como base del nuevo protocolo de comunicación. Un protocolo de comunicación utiliza los tipos para definir las transmisiones de datos. Luego no hubo la necesidad de inventar tipos de datos, sino que adoptaron los tipos de datos definidos en XML.

Luego de un año Dave Winer, creó su propio protocolo de comunicaciones, llamado XML-Llamada a procedimiento remoto (XML-RPC), el cual gano popularidad hasta la liberación de SOAP1.1 en el 2000.

SOAP 1.0 utilizaba HTTP como protocolo de transporte para los mensajes, aunque http es muy utilizado, su uso violaba uno de los objetivos claves de SOAP, la interoperabilidad. Este no debe depender de un protocolo de transporte. Con la presentación de SOAP 1.1, se hizo completamente independiente, lo que significaba enviar mensajes sin importar el protocolo, por ejemplo SMTP y FTP [2], además de HTTP.

SOAP 1.1 resuelve las preocupaciones de la industria de servicios Web. Sin embargo, no fue sino hasta el final del 2000 cuando el equipo de SOAP llamo la atención de la industria. Microsoft lo adopta como el protocolo de comunicaciones que se utilizaría en la tecnología .NET.

Se envía la definición de SOAP 1.1 al consorcio de la World Wide Web (W3C), organización que define los estándares de la comunidad Web. El W3C formo su propio comité, el cual incluía a Sun Microsystems, y se inicia el desarrollo de especificaciones de un protocolo basado en SOAP 1.1. El comité adopto un borrador de trabajo de la nueva especificación llamada SOAP1.2.

A finales del 2000 un grupo de empresas, incluyendo a Microsoft y a IBM, iniciaron esfuerzo por crear una comunidad de organizaciones que proporcionaran servicios Web. Dicha comunidad centra su atención alrededor de un grupo de registros en la Web que contiene información sobre cada miembro, sus servicios Web y las interfaces de programación de aplicaciones que se utilizan para obtener acceso a dichos servicios. Este grupo de registros se conoce *Descripción, descubrimiento e integración universal (UDDI).*

El 40% de los ingresos totales del mercado TI desde 2002 hasta 2006 provinieron de la tecnología WS y, a finales de 2005, las ventas de licencias de software que utilizaron sus estándares alcanzaron los 21.000 millones de dólares. Según

Gartner[2] mostraron la relevancia que tendrían en el futuro una tecnología incipiente e inmadura, pero en la que gigantes de la talla de IBM. Microsoft, HP y BEA System estaban invirtiendo miles de millones de dólares, para perfilarse como la panacea de la gran integración de aplicaciones en el seno de las empresas, al ser su implantación mucho más sencilla y barata, frente a la del software EAI (Enterprise Application Integration). [4]

El crecimiento no se encuentra en las computadoras sino en los dispositivos móviles conectados a la red, y el lenguaje de programación Java se mueve en ese sentido integrándose con los Servicios Web, afirmo Scott McNealy, presidente mundial de Sun Microsystems, en la octava conferencia JavaOne en el año 2003.

En Abril de 2004 el estándar WS-Security 1.0 fue publicado por Oasis-Open. En 2006 fue publicada la versión 1.1. Originalmente fue desarrollado por IBM, Microsoft, y VeriSign, el protocolo es ahora llamado oficialmente **WSS** y está desarrollado por un comité en Oasis-Open. El protocolo contiene especificaciones sobre cómo debe garantizarse la integridad y seguridad en mensajería de Servicios Web. El protocolo WSS incluye detalles en el uso de SAML y Kerberos, y formatos de certificado tales como X.509. WS-Security incorpora características de seguridad en el encabezado de un mensaje SOAP, trabajando en la capa aplicación. Así asegura seguridad extremo a extremo.

[2] **Gartner, S.A.** Es un proyecto de investigación de tecnología de la información y de firma consultiva con sede en Stamford, Connecticut. Se conocían como el grupo Gartner hasta 2001.

5 WEB SERVICES

Los servicios web son la revolución informática de la nueva generación de aplicaciones que trabajan colaborativamente en las cuales el software está distribuido en diferentes servidores**.** Los SW son considerados como un repositorio de servicios de ***n*** aplicaciones distribuidas por internet.

Un Servicios Web constituye un conjunto de protocolos y estándares, que sirven para intercambiar datos entre aplicaciones de software distintas desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de los servicios Web. Para mejorar la interoperabilidad entre distintas implementaciones de servicios Web se ha creado el organismo WS-I, encargado de desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares.

Los servicios Web aportan gran independencia entre la aplicación que usa el servicio Web y el propio servicio. De esta forma, los cambios a lo largo del tiempo en uno no deben afectar al otro. Esta flexibilidad será cada vez más importante, dado que la tendencia a construir grandes aplicaciones a partir de componentes distribuidos más pequeños es cada día más utilizada.

5.1 *WEB SERVICES SECURITY, CONCEPTOS BÁSICOS*

Si bien los beneficios aportados por los servicios Web al ámbito comercial son importantes (arquitectura descentralizada, independencia de la plataforma, disponibilidad, acceso abierto al público, facilidad y uniformidad en el acceso), los servicios Web traen consigo importantes riesgos desde el punto de vista de la seguridad que deben ser abordados desde diferentes niveles:

- Asegurar la autenticación bidireccional entre el cliente que accede y consume los servicios Web y el proveedor de los servicios accedidos y consumidos.

- Permitir la autorización del acceso a recursos y procesos en un contexto en el que deben poder ser administrados y controlados los accesos por parte de clientes, proveedores, vendedores, competidores, intrusos.

- Identificación y autentificación única de los usuarios de modo que puedan acceder a diversos sistemas sin necesidad de autenticación en cada uno de ellos.

- Garantizar la confidencialidad de los datos intercambiados entre el cliente y el servidor. SOAP no cifra la información, luego viaja en texto plano por Internet. Garantizar la integridad de los datos para protegerlos de alteraciones (accidentales o intencionadas).
- Impedir el repudio de las operaciones.

- Protección frente a los ataques típicos de otras tecnologías. Dado que los servicios Web transcurren por los puertos 80 y 443, los filtrados de puertos clásicos poco pueden hacer frente a los ataques a nivel de aplicación. Sin embargo se puede hacer uso de analizadores de tráfico de nivel de aplicación, que entienden los servicios y son capaces de interceptar posibles ataques a las aplicaciones.

Las tecnologías basadas en XML han supuesto un importante cambio en la forma de pensar de los arquitectos de sistemas. Gracias a las capacidades de interoperatibidad, distribución, integración e independencia de las plataformas y tecnologías subyacentes a las aplicaciones, XML se ha consolidado como el estándar para el intercambio de datos.

El tiempo de penetración de ésta tecnología ha sido increíblemente más bajo que el de cualquier otra que se haya encontrado hasta el momento, y sus aplicaciones están avanzando día a día para la descripción de contenidos, desde simples representaciones de registros existentes en bases de datos a la representación de mundos de realidad virtual, la descripción de contenidos multimedia, los nuevos estándares de codificación de contenidos Web, los servicios Web para el intercambio de información entre aplicaciones.

En WS se trabaja bajo el régimen del esquema XML (*XML Schema*), creados por una herramienta de desarrollo y puestos dentro del WSDL del Web Services.

Todos los estándares usados para la descripción, descubrimiento e invocación de los servicios son basados en XML. SOAP y WSDL son descritos por XML Schema. La estructura básica del estándar de seguridad, XML Encryption, XML Signature, Security Assertion Markup Language(SAML), and WS-Security son basados en XML y descritos por XML Schema.

Las organizaciones pueden ofrecer Servicios Web a otras organizaciones mediante el uso de un conjunto de estándares basados en registros XML. Para interactuar con estos registros, una aplicación puede utilizar la API de Java, ***JAXR***, la cual contiene todas las clases e interfaces necesarias para consultar o publicar un servicio en un registro XML.

WS-Security proporciona un conjunto amplio de dispositivos de seguridad para aplicaciones de servicios Web, al basarse en estándares establecidos de la industria respecto a criptografía y cifrado y firmado de XML. Puede especificar los dispositivos a utilizar en una aplicación específica con WS-Policy y WS-SecurityPolicy, lo que permite que los clientes del servicio se configuren automáticamente para acceder al servicio. Con el soporte generalizado de esos estándares en varias plataformas e infraestructuras de SW, hay una buena interoperatividad (que mejora a lo largo del tiempo).

5.1.1 XML EXTENSIBLE MARKUP LANGUAGE

XML fue creado como una estructura que describe a sí misma la manera de representar los datos siendo totalmente independiente de aplicaciones, protocolos, vocabulario, sistemas operativos o incluso lenguaje de programación. XML puede ser considerado el lenguaje franco de los negocios debido a que está siendo utilizado ampliamente en todas las industrias ya que la transmisión de datos del negocio se hace de manera portable. El uso de XML presenta grandes desafíos en el tema de la seguridad.

El XML proporciona un marco de trabajo para etiquetado de datos estructurados; A medida que se adoptan las etiquetas XML a lo largo de una intranet de alguna organización y por otros a lo ancho de la Internet, habrá una correspondiente habilidad para buscar y manipular datos sin importar las aplicaciones dentro de las cuales se encuentre. Una vez que los datos han sido localizados, pueden ser transferidos a través de la red y presentados en un navegador tal como el Internet

Explorer en una gran variedad de formas, o puede ser transferido a otras aplicaciones para su posterior procesamiento y visualización.

Los datos destinados para un WS pueden ser creados en XML o convertidos en XML desde un formato nativo. Estos datos pueden ser tomados desde tablas pertenecientes a una base de datos relacional o procesadas por un lenguaje de programación tal como Java o C++ y luego transformados a XML.

La estructura de un archivo **XML** es muy similar a la **estructura** en HTML. La diferencia está en nombrar las etiquetas, almacenando la información que nos interesa y con lo cual se logre acceso a esta en el momento que se requiera.

5.1.2 SOAP (SIMPLE OBJECT ACCESS PROTOCOL)

Es un protocolo de comunicación que facilita la interacción entre aplicaciones y servicios Web ubicados en máquinas remotas. SOAP permite la separación impecable entre el procesamiento de la infraestructura y el procesamiento de la aplicación de mensajes.

Los datos y el mensaje no se almacenan en el transcurso de la transmisión. No se puede hacer referencia a objetos, ya que todos los datos deben estar contenidos en forma explícita, lo que significa que el mensaje no puede contener referencias a datos externos.

5.1.2.1 TRANSPORTE DE MENSAJES

SOAP no impone el uso de un determinado protocolo para el intercambio de mensajes. A través del concepto de “binding” SOAP permite especificar:

- cómo los mensajes SOAP se encapsulan en un protocolo de transporte.
- cómo los mensajes SOAP deben ser tratados con las primitivas del protocolo.

En el estándar SOAP se incluye la especificación de un “*binding*” para el protocolo HTTP. Como función implícita permite el direccionamiento de los mensajes. La forma de especificar el WS destino, depende entonces del protocolo de transporte utilizado, por ejemplo:

- HTTP: URL del recurso destino.
- SMTP: la dirección “to” en el header del e-mail.

5.1.2.2 PROCESAMIENTO DE MENSAJES

En el modelo SOAP pueden existir varios nodos intermediarios que procesan los mensajes.

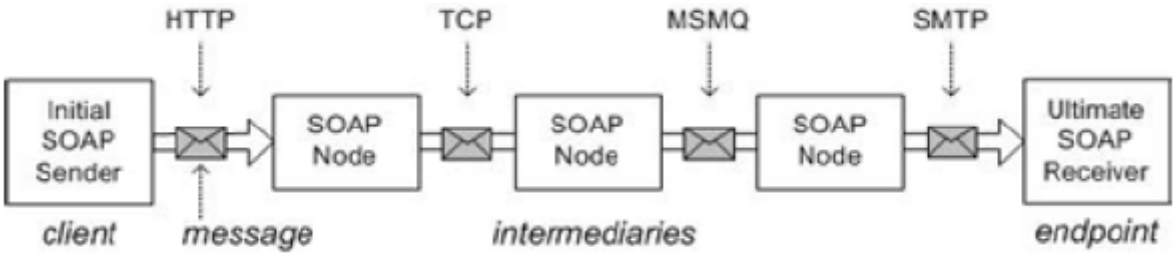


Ilustración 3 Procesamiento de mensajes SOAP

Los “header blocks” (bloques de encabezado) pueden incluir información que especifique para qué rol está destinado el bloque (atributo role). SOAP define tres posibles valores para dicho atributo: *none (ninguno), ultimateReceiver (Receptor Final) y next (Siguiente)*. SOAP no incluye un mecanismo para especificar la ruta que debe seguir un mensaje SOAP hasta llegar al WS destino.

5.1.3 WSDL Y UDDI

***UDDI* (Universal Description, Discovery and Integration)** es un registro público diseñado para almacenar de forma estructurada información sobre empresas y los servicios que éstas ofrecen. A través de UDDI, se puede publicar y descubrir información de una empresa y de sus servicios. Se puede utilizar sistemas taxonómicos estándar para clasificar estos datos y poder encontrarlos posteriormente en función de la categorización.

Lo más importante es que UDDI contiene información sobre las interfaces técnicas de los servicios de una empresa. A través de un conjunto de llamadas a **API** [5] XML basadas en SOAP, se puede interactuar con UDDI tanto en tiempo de diseño como de ejecución para descubrir datos técnicos de los servicios que permitan invocarlos y utilizarlos. De este modo, UDDI sirve como infraestructura para una colección de software basado en servicios Web.

Varias empresas, incluidas Microsoft, IBM, Sun, Oracle, COMPAQ, Hewlett Packard, Intel, SAP entre otras, unieron sus esfuerzos para desarrollar una especificación basada en estándares abiertos y tecnologías no propietarias. El resultado, fue la versión beta lanzada en diciembre del 2000 y llevada a producción en mayo de 2001, fue un registro empresarial global alojado por varios nodos de operadores en el que los usuarios podían realizar búsquedas y publicaciones sin coste alguno.

A partir de la creación de esta infraestructura para servicios Web, los datos sobre estos servicios se pueden encontrar de forma sistemática y confiable en una capacidad universal totalmente independiente de proveedores. Se pueden llevar a cabo búsquedas categóricas precisas utilizando sistemas de identificación y taxonómicos extensibles. La integración de UDDI en tiempo de ejecución se puede incorporar a las aplicaciones. Fomentando el desarrollo de un entorno de software de servicios Web.

Por otra parte, **WSDL (Web Services Description Language)** se ha convertido en una pieza clave de la pila de protocolos de los servicios Web. WSDL y UDDI se diseñaron para diferenciar claramente los metadatos abstractos y las implementaciones concretas. Para entender cómo funcionan WSDL y UDDI resulta esencial comprender las consecuencias de esta división. Por ejemplo, WSDL distingue claramente los mensajes de los puertos: los mensajes (la sintaxis y semántica que necesita un servicio Web) son siempre abstractos, mientras que los puertos (las direcciones de red en las que se invoca al servicio Web) son siempre concretos.

No es necesario que un archivo WSDL incluya información sobre el puerto, simplemente contener información abstracta de interfaz, sin facilitar datos de implementación concretos, y ser válido. De este modo, los archivos WSDL se separan de las implementaciones.

Pueden existir varias implementaciones de una única interfaz WSDL. Este diseño permite que sistemas dispares escriban implementaciones de la misma interfaz, para garantizar así la comunicación entre ellos. Si tres empresas diferentes implementan el mismo archivo WSDL y una parte del software de cliente crea el código auxiliar/proxy a partir de esa interfaz, dicho software se podrá comunicar con las tres implementaciones con el mismo código de base, cambiando simplemente el punto de acceso.

UDDI establece una distinción similar entre la abstracción y la implementación con el concepto de tModels. La estructura tModel, abreviatura de "Technology Model" (modelo de tecnología), representa huellas digitales técnicas, interfaces y tipos

abstractos de metadatos. El resultado de los tModels son las plantillas de enlace, que son la implementación concreta de uno o más tModels. Dentro de una plantilla de enlace se registra el punto de acceso de una implementación particular de un tModel.

Del mismo modo que el esquema de WSDL permite separar la interfaz y la implementación, UDDI ofrece un mecanismo que permite publicar por separado los tModels de las plantillas de enlace que hacen referencia a ellos. Por ejemplo, un grupo industrial o de estándares publica la interfaz canónica para un sector particular y, a continuación, varias empresas escriben implementaciones de esta interfaz. Obviamente, cada una de estas implementaciones haría referencia al mismo tModel.

Los archivos WSDL constituyen un ejemplo perfecto de tModel de UDDI.

5.2 PROTOCOLOS EN APLICACIONES WEB

El modelo de seguridad, identifica y evalúa las vulnerabilidades de un sistema y los riesgos a los que se exponen, con el fin de establecer políticas y mecanismos de carácter preventivo, detectivo y correctivo que garanticen la seguridad del mismo.

Inicialmente, se analiza e identifica los riesgos de seguridad a los que se expone un WS cuando es publicado. Es importante definir para cada riesgo un control preventivo, detectivo y correctivo. Finalmente, se establecen niveles de seguridad y se analiza la relación costo/beneficio de su implementación.

Algunos de los riesgos a los que se enfrenta un WS a causa de sus vulnerabilidades se identifican a continuación:

- Acceso no autorizado a la información, recursos y servicios.
- Falta de mecanismos de validación de la identidad de las aplicaciones que acceden al servicio en representación de un cliente gestor.

- Interferencia en el canal de comunicación establecido entre las entidades participantes de forma activa (alterar el contenido del mensaje) o pasiva (conocer el contenido del mensaje).

- Suplantación de la entidad de una o varias entidades.

- Repudiación o no uso de los registros de acciones del sistema (LOG´s).

- Infección por virus en cualquiera de las entidades o máquinas contenedoras.

- Ausencia de definición de requerimientos, estándares, controles y/o políticas de seguridad del sistema.

- Ausencia de controles en la definición de documentos de descripción de servicio por parte de la entidad de seguridad y proveedora.

- Incorrecta definición, implementación, configuración y control de la entidad de seguridad.
- Emisión de tokens de seguridad sin tiempo límite de Vigencia.

- No ejecución de un plan riguroso de pruebas sobre el sistema de WS.

- Interrupción de uno o varios servicios prestados a causa de falta o fallas del fluido eléctrico.

- Funcionamiento incorrecto de servidores (contenedor del WS) y dispositivos de redes.

- Rutinas de código malignas existentes en dispositivos de redes como servidores o equipos que afectan directamente al sistema de WS.

- Ausencia de planes de contingencia.

Por lo anterior, se hace precisa la creación de servicios seguros encargados de la administración y emisión de tokens de seguridad con la finalidad de garantizar la *autenticación* de usuarios en el sistema y crear un contexto federado que establece un conjunto de participantes con relaciones de confianza implícitas entre ellas.

WSDL puede ser inocentemente el más grande agujero en el despliegue de los Web Services. Las herramientas de desarrollo generan el archivo WSDL en una

localización estándar. De manera impecable es públicamente anunciado al mundo exterior el *what, how y where* del servicio. Cualquier persona puede leer y comenzar a acceder el Servicio Web. Si no existe una seguridad adicional tal como requerir autenticación, entonces cualquiera fácilmente podría acceder a la información que genera el WS.

El Servicio Web podría proveer esencialmente acceso directo al núcleo de las aplicaciones empresariales o a la base de datos, por tanto la urgencia de proteger la WSDL URL con certificados SSL, con los que el servidor niega el acceso a cualquiera que no sea identificado.

Hacemos relación a cada componente de la familia WS-* a uno o más principios de seguridad; con el fin de establecer que componente y como el modelo daría soporte a cada uno de los sietes principios con los avances realizados hasta el momento por otras organizaciones.

Uno de los principales propósitos de *WS- Security* provee un modelo unificado que utiliza las tecnologías existentes y que permite que las aplicaciones intercambien mensajes SOAP de forma segura. El mismo provee mecanismos extensibles y flexibles, pero esta extensibilidad puede afectar la interoperabilidad de las distintas plataformas tecnológicas.

La **WS-I (*Web Services Interoperability Organization*),** define cómo se deberían cumplir los requerimientos de interoperabilidad al utilizar un conjunto de tecnologías que aseguran la transmisión de los mensajes SOAP. Según la WS-I los desafíos de seguridad en la transmisión mensajes SOAP son los siguientes:

- Identificación y Autenticación de las partes.
- Identificación y Autenticación de los datos de origen.
- Integridad de los datos: Integridad en el transporte y del mensaje SOAP.
- Confidencialidad de los datos: Confidencialidad en el transporte y del mensaje SOAP.
- Unicidad del mensaje SOAP.

La capa base de SOAP se compone de tecnologías clave, como SOAP, WSDL, XML Signature, XML Encryption y SSL / TLS que son obtenidos por la especificación del *WS-Security*. Las especificaciones construidas sobre el *WS-Security*, se describen brevemente como sigue:

- **WS-POLICY**

  Así como el nombre lo implica, *WS-Policy* permite a una organización que expone un *Web Services* especificar una política para ese servicio. La política detalla que capacidades y limitaciones presenta un servicio en materia de seguridad, por ejemplo, qué tipo de algoritmo cifrado soporta y que parámetros necesitan para la encriptación.

  La política puede ser utilizada para descubrir de un servicio sus capacidades y limitaciones en materia de seguridad, similar a cómo WSDL se utiliza para describir las características generales de un servicio (O'Neil, 2003).

- **WS-TRUST**

  Describe como las relaciones de confianza son creadas, direct o brokered. En el caso de los *Web Services* y la seguridad, la confianza significa que se puede confiar en relación con la autenticidad y la autorización de una entidad si un tercero en que puedes confiar da fe de ello. Ejemplos comunes de confianza son los modelos de PKI (modelo de confianza jerárquica) y PGP (un modelo de red de confianza).

  El modelo *WS-Trust* también permite que la delegación y la suplantación se lleven a cabo. Esto significa que un mensaje SOAP no es enviado directamente al usuario final, sino que un programa de software actúa en su nombre, un *token* de seguridad que contiene información se retorna de vuelta al usuario final insertado en el mensaje.

  Este *token* se presenta a los controles de seguridad como lo que necesita el mensaje para ser autenticado a lo largo de su camino. De este modo, el usuario final es impersonado mediante la información que figura en el token. La delegación y la suplantación son esencialmente compatibles con

cualquier otro, salvo que la delegación ofrezca posibilidades adicionales relativas a la trazabilidad (IBM Y Microsoft, 2002).

- **WS-PRIVACY**

La información relativa a un usuario final y el envío de esta información con el mensaje, ya que esta es ruteada entre diferentes nodos pueden tener un impacto en el usuario final y la protección de la intimidad en función de que tan delicada sea la información.

La especificación de *WS-Privacy* hace que sea posible para un proveedor de servicios expresar su política en relación con cuestiones de privacidad y para el prestador de servicios exigir que las solicitudes especificadas del remitente se adhieran a esa política.

Para lograr este objetivo, el *WS-Privacy* a su vez, hace uso de una combinación de otras especificaciones, incluyendo *WS-Policy*, *WS-Security* y *WS-Trust*.

- **WS-SECURE CONVERSATION**

Esta especificación tiene por objeto hacer posible la creación de sesiones que abarcan varios mensajes SOAP por lo que no será necesario evaluar cada mensaje entrante con respecto a la autenticación y autorización. El *WS-Security* no es compatible con las sesiones que pueden dar lugar a problemas de rendimiento. O'Neil (2003) describe *WS-Secure Conversation* como "SSL en el nivel SOAP" porque probablemente SSL se presenta como un modelo para la forma de dirigir y establecer la clave de sesión.

SSL utiliza el más lento cifrado de clave pública de tecnología segura para el intercambio de una reunión clave entre el cliente y el servidor. Una vez que esta clave se ha establecido, un esquema de cifrado mucho más rápido de clave privada puede ser utilizado durante todo el período de sesiones para cifrar el tráfico de datos.

- **WS-FEDERATION**

Debido a que diferentes partes implicadas en el consumo y la prestación de un servicio pueden utilizar diferentes tecnologías de seguridad, por ejemplo, una de las partes podrá utilizar Kerberos y otro Certificados X.509, puede ser necesario traducir los datos relativos a la seguridad entre las partes implicadas. *WS-Federation* es la especificación que describe la forma en que la intermediación entre las partes debe llevarse a cabo. La especificación funciona en una capa por encima de *WS-Policy* y *WS-Trust*.

- **WS-AUTHORIZATION**

Esta especificación se refiere a la manera de expresar las reglas con respecto a lo que se permite el acceso y cómo estas normas pueden ser manejadas. Más concretamente, la especificación describe cómo los *claims* son representados por *tokens* y cómo estos deben interpretarse en el endpoint.

- **SAML**

(*Security Assertion Markup Language*), provee un marco de trabajo para intercambiar información en forma segura. La especificación SAML 2.0 define los protocolos y perfiles de identidad para que la federación dentro de las limitaciones de los diferentes casos de uso. Además, la especificación de SAML 2.0 define un *token* de seguridad llamado *SAML assertion*.

Como un token de seguridad, el *SAML assertion* tiene muchas propiedades únicas que lo hacen muy útil para la federación de identidades. Es un estándar abierto, utiliza la sintaxis XML en un documento XML Soporta entornos heterogéneos, es flexible y extensible, transmite la autenticación, autorización y atributo de información, opcionalmente, se auto valida (self-validating).

- **XACML**

(Extensible Access Control Markup Language), es un lenguaje para escribir políticas de control de acceso.

- **XKMS**

(XML Key Management Specification), es un mecanismo basado en XML para la gestión de una Infraestructura de Clave Pública (PKI).

Permitiendo firmar documentos XML, Partes de documentos XML y Objetos no XML (por ejemplo. Imágenes).

5.2.1 SEGURIDAD EN EL NIVEL DE TRANSPORTE

La seguridad en el nivel de transporte consiste en aplicar las características que mejoran la seguridad existente que esté siendo usado por el servicio Web en cuestión. Esto es posible gracias a la encapsulación que tiene lugar en la pila de protocolos, desde el nivel de aplicación al nivel de transporte.

En lo referido a seguridad para servicios Web, la seguridad a nivel de transporte se emplea para establecer una seguridad punto a punto: la comunicación es segura desde un punto a otro punto, lo que implica que la conexión sea directa, sin intermediarios. Para lo relativo a servicios Web, intermediarios como los enrutadores de Internet no se consideran intermediarios, ya que estos han sido diseñados para soportar la seguridad a nivel de transporte.
|

Cabe señalar que tecnologías que aportan seguridad a nivel de transporte también pueden ser empleadas a niveles inferiores, como el nivel de red. *IPSec*[3] es un ejemplo de tecnología que podemos usar para asegurar la confidencialidad y en cierto modo la autenticidad de los consumidores y prestadores de servicios Web.

[3] (abreviatura de Internet Protocol security) es un conjunto de protocolos cuya función es asegurar las comunicaciones sobre el Protocolo de Internet (IP) autenticando y/o cifrando cada paquete IP en un flujo de datos. Incluye protocolos para el establecimiento de claves de cifrado.

Podemos aseverar que en lo relativo a servicios Web en los que no existen intermediarios (por ejemplo intermediarios de SOAP), los mecanismos de seguridad subyacentes al nivel de aplicación son suficientes para mantener un grado de confidencialidad aceptable. Por otra parte, si lo que perseguimos es una seguridad de extremo a extremo, aunque se haga uso de intermediarios de SOAP, se debe recurrir a las técnicas de seguridad propias del nivel de aplicación, incluidas nuevas tecnologías propias de XML.

Ilustración 4 Comunicación extremo a extremo

La seguridad implementada en el nivel de transporte no incide en ningún aspecto sobre el mensaje SOAP, sin embargo requiere de modificaciones en las configuraciones de los servidores, los códigos de los clientes para hacer las llamadas apropiadas para la autenticación y algunas soluciones en el hardware de la red.

5.2.2 SEGURIDAD A NIVEL DE APLICACIÓN

5.2.2.1 SEGURIDAD A TRAVÉS DE HTTP

El protocolo de transferencia de hipertexto es quizá el protocolo más conocido en Internet, y a través de él circulan la mayor parte de los contenidos Web. Como protocolo, su comportamiento es síncrono (cuando se envía una petición, hay que recibir una respuesta a la misma antes de poder continuar). Las peticiones se realizan envueltas en unas cabeceras del protocolo HTTP, y las respuestas tienen un formato muy similar a las peticiones. Ambas viajan en texto plano por Internet.
El protocolo HTTP carece de estados, por lo que implica que cada par de petición y respuesta es un acto individual, sin relación teórica con los demás. Existen

diferentes maneras en las que se ha dotado al protocolo HTTP de comportamiento de sesión emulado (bien mediante cookies, registros de autentificación del navegador,…) pero en cualquier caso no forman parte de la definición del protocolo, sino que son soluciones aportadas y asumidas por los servidores y los clientes. La comunicación sin estado tiene como ventaja la facilidad para adaptar sistemas de distribución de mensajes.

El protocolo HTTP, en su versión 1.1 establece dos tipos de autenticación para restringir el acceso del cliente a los servicios:

- **AUTENTICACIÓN BÁSICA**

  Es un sencillo protocolo de identificación definido en la especificación del protocolo HTTP en su versión 1.0. Todos los servidores Web y navegadores incluyen la autenticación básica. La autenticación básica funciona de forma fluida a través de casi todos los cortafuegos y proxies; los toolkits de SOAP incluyen este esquema de autenticación también.

  Sin embargo, el aspecto negativo de la autenticación básica es que envía las contraseñas en el formato de codificación Base64, que es reversible y muy simple. Por ello, resulta completamente necesario el uso asociado de la autenticación básica en combinación con **SSL (Secure Sockets Layer).**
- **AUTENTICACIÓN DE COMPENDIO (DIGEST)**

  Su funcionamiento es similar a la autenticación básica, pero su uso no está tan extendido. No obstante está soportada por los principales servidores Web del mercado y los navegadores. La principal diferencia con la autenticación básica de HTTP es que nunca se envía el nombre de usuario ni la clave en la cabecera HTTP, sino un resumen de la misma obtenido mediante el algoritmo MD5 aplicado sobre el usuario, la clave y un valor "*nonce*" enviado por el servidor a la hora de solicitar la autenticación para el acceso a un recurso.

  El valor nonce queda libre para la elección por parte del fabricante del servidor, pero es sugerido por el estándar usar una combinación de la dirección IP del cliente, el URI al que accede y una marca de tiempo. De esta manera se amplía el espectro de claves, y un ataque de diccionario contra el resumen expuesto empieza a perder el sentido con la tecnología disponible en el momento actual.

Este enfoque nos puede proveer de una falsa sensación de seguridad al ser cierto que, aunque un atacante que intercepte el tráfico entre el cliente y el servidor no será capaz de conocer el usuario ni la clave. Sin embargo, el método de autenticación por compendio es frágil ante ataques de retransmisión de los credenciales.

Los mecanismos de autenticación provistos por HTTP no proporcionan confidencialidad de la información transmitida, simplemente limitan el acceso de los usuarios no autorizados a los recursos no autorizados. Desde el punto de vista de la seguridad, el mecanismo de autenticación por compendio es mucho más deseable que la autenticación básica (que debería tender a desaparecer en beneficio de todos). Sin embargo es patente que los mecanismos de autenticación provistos por HTTP se encuentran en el nivel más bajo de seguridad, desde un punto de vista criptográfico.

Es obvia la necesidad de combinar los mecanismos de autenticación a nivel de aplicación con otras tecnologías que nos aporten confidencialidad e integridad en la información. Una alternativa elegible para fortalecer los mecanismos de autenticación expuestos es la combinación de los mismos con SSL. Cuando aplicamos SSL sobre HTTP obtenemos HTTPS (o HTTP asegurado), que nos asegura confidencialidad extremo a extremo en las conexiones sin intermediarios, y confidencialidad punto a punto en las conexiones con intermediarios SOAP.

Otros mecanismos de autenticación que presentan debilidades similares a los provistos por el protocolo HTTP incluirían la autenticación por formulario en la aplicación Web (en la que una parte de la aplicación se responsabiliza de identificar y autenticar al usuario por las credenciales presentadas) o la autenticación integrada en Windows /NT Challenge Response (NTLM).

Éste último es una implementación propia de Microsoft, análoga a la autenticación por compendio, pero solo soportada por sus servidores y sus navegadores. Además de ofrecer los mismos problemas a los mecanismos comentados anteriormente, se añade la dificultad de no estar implementados en todos los posibles sistemas que accedan a consumir los servicios ni en los servidores (lo que termina con la independencia de la plataforma en la implementación del acceso a servicios Web).

5.2.2.2 ENVÍO DE CREDENCIALES EN MENSAJES SOAP

Al aplicar seguridad en el nivel de aplicación, podemos modificar el propio mensaje SOAP para que incorpore los credenciales del usuario que quiere consumir el servicio.

Este enfoque requiere que tanto el servicio como el cliente incorporen el mecanismo de autenticación para poder ofrecer el acceso al servicio. La aplicación de este método haría viajar los credenciales en texto plano, a no ser que nos encontremos usando el protocolo SSL para la provisión del servicio Web. En cualquier caso, este enfoque es análogo a la autenticación basada en formulario comentada en el apartado anterior. Admite varias maneras se ser llevado a cabo:

- Enviar los credenciales en cada mensaje SOAP transmitido, de modo que el servidor pueda validarlos para cada petición.

- Enviar los credenciales en el primer mensaje SOAP transmitido, de domo que el servidor pueda validarlos y establecer internamente una sesión para el usuario. El servidor devuelve al usuario un identificador de sesión que el cliente podrá usar en cada petición posterior, y el servidor simplemente comprobar si la sesión fue establecida previamente, y si el acceso al recurso solicitado es concedido al usuario propietario de la sesión indicada. Con la finalidad de aumentar la seguridad en este enfoque, es recomendable establecer periodos de caducidad para las sesiones.

Es importante remarcar que este modelo de autenticación debe ir siempre acompañado del protocolo SSL para evitar que los credenciales se envíen en texto plano. Establecer un esquema de autenticación que no preserve la confidencialidad del cliente, la integridad de la información y que permita la suplantación de personalidad en un canal no cifrado puede llegar a ser más costoso computacionalmente que implementar los servicios Web a través de HTTPS, y nunca cubriría los requisitos mínimos de confidencialidad con las tecnologías presentadas hasta el momento.

5.2.2.3 AUTENTICACIÓN KERBEROS Y BASADA EN TICKETS

La autenticación basada en tickets tiene lugar en el nivel de aplicación, lo que significa que se puede utilizar con cualquier protocolo de transporte. Por lo

general, la autenticación basada en tickets usa firmas digitales, certificados digitales y tecnologías de codificación de clave pública. Dentro de una aplicación podemos incluir código para llevar a cabo la autenticación, la codificación del contenido de los mensajes, firmarlos digitalmente o validar las firmas de los mensajes recibidos.

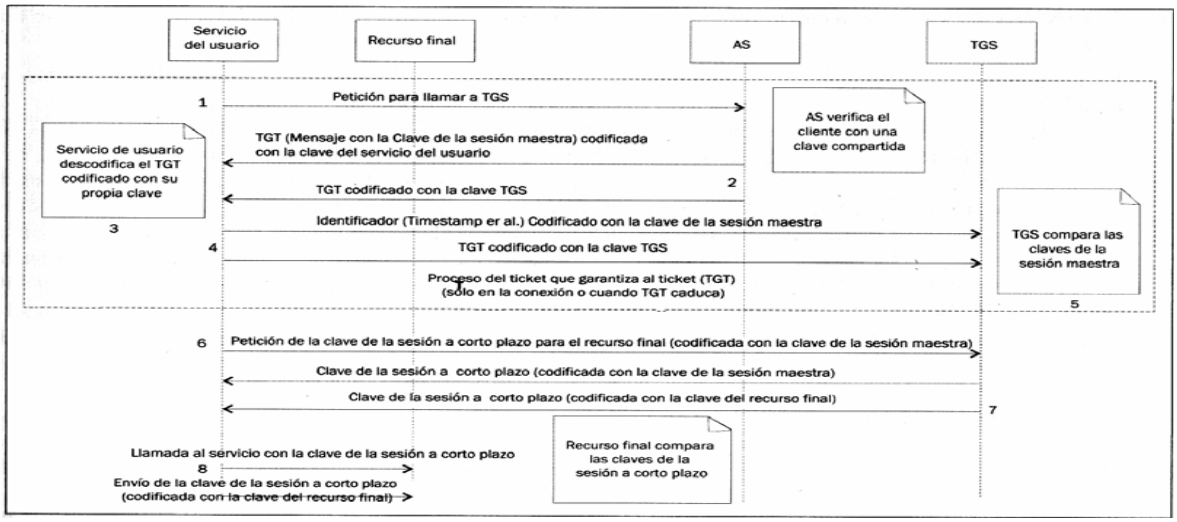


Ilustración 5 Diagrama de Secuencia de Autenticación Kerberos

La autenticación basada en tickets, implica un mayor esfuerzo en el desarrollo de las aplicaciones, sin embargo la flexibilidad que proporciona es elevada y la seguridad aportada es bastante superior a los esquemas vistos para la autenticación en el protocolo HTTP o en los mensajes SOAP. Compañías como Microsoft está haciendo uso de la autenticación basada en tickets para los servicios de autenticación (Microsoft Passport) de una sola conexión (single sign on) y sus servicios Hailstorm[4].

Kerberos se desarrolló en el MIT[5], con el objetivo de ser un servicio de autenticación. Su punto de partida es que solo un número limitado de máquinas pueden ser muy seguras, por lo que trata todas las máquinas de la red como inseguras, a excepción de una de ellas. Kerberos permite la autenticación mutua entre usuarios y servicios, y dispone de mecanismos de delegación.

El mecanismo de funcionamiento de Kerberos es simple. La ***figura 5*** presenta el diagrama de secuencia de Kerberos, y a continuación se describe los pasos, asociados a la numeración de la figura:

[4] Aplicación Microsoft
[5] Instituto Tecnológico de Massachusetts

1. Antes de acceder a cualquier servicio, el servicio del usuario solicita un ticket para contactar con el Ticket Granting Server (TGS) como si fuera cualquier otro servicio. Esta petición es enviada al Authentication Service (AS).

2. El AS crea una clave de sesión maestra y hace dos copias de la misma. Coloca una copia en un mensaje y lo codifica con la clave que pertenece al servicio de usuario. Este mensaje es el Ticket Granting Ticket (TGT). El AS coloca la otra copia de la clave de la sesión en otro mensaje que codifica con una clave que pertenece al TGS y envía ambas copias al servicio de usuario.

3. El servicio de usuario recibe los dos mensajes del AS. Puede abrir el mensaje que ha sido codificado con su clave, pero no puede leer el mensaje codificado con la clave del TGS.

4. El servicio del usuario crea un mensaje nuevo y coloca una marca de tiempo que servirá como una clave de sesión adicional y lo codifica con la clave de sesión maestra. Coge el mensaje recibido del AS y lo cifra con la clave de TGS, junto con el nuevo mensaje de autentificación adicional que contiene la marca de tiempo. Finalmente envía el mensaje al TGS.

5. El TGS recibe los dos mensajes y abre el que contiene la clave de sesión maestra (solo TGS puede abrir este mensaje, que envió originalmente el AS). El TGS usa el segundo mensaje que envió el servicio de usuario y lo descifra usando la clave maestra de sesión. Obtiene la información de autentificación adicional añadida por el servicio de usuario (timestamp, fecha, hora…).

   Con ello el TGS entiende la identidad del servicio del usuario. Ahora, el servicio del usuario y el servicio que otorga el ticket se pueden comunicar el uno con el otro usando la clave de la sesión maestra.

6. El servicio de usuario solicita un ticket de corto plazo al TGS para contactar con cualquier recurso de la red.

7. El TGS contesta con una clave de sesión de corto plazo para acceder al servicio de recurso de destino. Esta clave de corto plazo está codificada con la clave de sesión maestra que se encuentra en el TGT, no con la clave del servicio de usuario.

   El TGS envía otra copia de la clave de sesión de corto plazo al cliente, pero codificada con la clave pública de recursos que presta el servicio. La clave de sesión de corto plazo tiene un tiempo de vida muy breve, incluso de un solo uso.

   El propio TGT tiene un periodo de vida muy corto, normalmente de 8 horas, de modo que si es robado, se puede reemplazar muy fácilmente y su periodo de validez es muy reducido (normalmente las contraseñas no se cambian con tanta frecuencia, y si son robadas, el compromiso que supone para los sistemas es mucho mayor).

8. Ahora el servicio del usuario puede contactar con el recurso de destino usando la clave de la sesión a corto plazo y, también, le envía el ticket codificado con la clave pública del recurso. El recurso final descodifica este ticket para obtener la clave a corto plazo, que usa a su vez para descodificar la petición desde el servicio del usuario.

Si tenemos en cuenta que el TGT y otros mensajes codificados que pueden ser enviados por los servidores se pueden intercambiar por medio de bloques de XML, parece claro que los tickets se pueden intercambiar a través de la Web para facilitar la autenticación en el acceso a servicios Web.

Como podemos deducir del funcionamiento de Kerberos, su uso para la implementación de seguridad en los servicios Web nos proporciona un alto grado de integridad, confidencialidad y autenticidad. El uso de Kerberos en XML requiere del uso de XML Encryption y de XML Signature que veremos más adelante.

Existe un XML-Schema que define el tipo de documento que se debe emplear para la autenticación Kerberos basada en XML.

5.2.3 TECNOLOGÍAS PARA SEGURIDAD INHERENTES A XML

A continuación veremos especificaciones que se han desarrollado para dotar los propios documentos XML de mayor seguridad, usando XML. También veremos la especificación de privacidad P3P, ya que la privacidad es un tema candente en el panorama actual, e incide en cierto modo sobre la seguridad.

5.2.3.1 XML SIGNATURE

XML Signature es un estándar desarrollado por el W3C que nos permiten autenticar al remitente, asegurar la integridad de partes de los documentos XML transportados. Puede ser aplicado a cualquier tipo de contenido digital (objetos de datos binarios), incluyendo documentos XML.

A través de la firma digital nos permite ofrecer garantía de autenticidad de los datos, la identificación de la identidad del emisor del mensaje, la validación de autenticidad e integridad del mensaje y, derivado de estas características, la propiedad de no repudio en los mensajes.

```
<Signature Id="EjemploXMLSignature"
 xmlns="http://www.w3.org/2000/09/xmldsig#">
   <SignedInfo>
     <CanonicalizationMethod
      Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
     <SignatureMethod
      Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
     <Reference
        URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126/">
        <DigestMethod
          Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>j6lwx3rvEPOOvKtMup4NbeVu8nk=</DigestValue>
      </Reference>
   </SignedInfo>
   <SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
   <KeyInfo>
      <KeyValue>
         <DSAKeyValue>
           <p>...</p><q>...</q><g>...</g><y>...</y>
         </DSAKeyValue>
      </KeyValue>
   </KeyInfo>
</Signature>
```

Ilustración 6 Ejemplo de documento XML Signature

Existe un elemento "*Signature*" que encapsula la firma digital dentro del documento. Contiene tres sub-elementos: *SignedInfo*, *SignatureValue* y *KeyInfo.* El elemento **SignedInfo** contiene información que describe qué es lo que se firma y cómo se lleva a cabo esa firma; gracias a esta información, la firma puede ser creada de nuevo para verificar su validez. Contiene a su vez la descripción de dos algoritmos, y las referencias a los objetos que se van a firmar.

El algoritmo descrito por el elemento ***CanonicalizationMethod*** se aplica para canonizar[6] el elemento SignedInfo, y se aplicará el primero en el orden de la generación de la firma. La canonización del documento puede llevarse a cabo de dos formas: incluyendo los comentarios u omitiéndolos del documento XML.

El elemento ***SignatureMethod*** indica el algoritmo que será empleado para la obtención de la firma digital a partir del objeto canonizado después de aplicar el método de canonización. En la gráfica se muestra que el documento presentará una firma generada mediante el algoritmo DSA (basado en DSS).
Otra posibilidad de algoritmo de firma recogida por el estándar es PCKS-1(basado en RSASSA-PKCS1-v1_5). El resultado obtenido de la aplicación del algoritmo de firma, debidamente representado, se incluye dentro del elemento SignatureValue codificado en Base64.

Cada elemento ***Reference*** que encontremos en el elemento SignedInfo incluye una referencia al objeto que será firmado. El elemento *Referenc*e incluye un elemento ***DigestMethod*** que indica el algoritmo de firma digital que se aplicará al objeto referenciado por este elemento. También incluye un elemento *DigestValue* que es el valor resultante de su firma.

El elemento *Reference* también puede incluir un elemento ***Transform*** opcional, que indique las transformaciones que se han aplicado al objeto antes de obtener valor de la firma.

El elemento ***SignatureValue*** contiene la firma digital asociada al elemento SignedInfo del documento. El elemento KeyInfo es un elemento opcional que se puede incluir en el documento para indicar la clave que se ha de usar para validar

[6] La forma canónica de un documento XML es una manera de mostrar una representación física que asegura que si dos documentos XML son aparentemente diferentes, pero sus canónicos son iguales, los dos documentos son equivalentes.

la firma. El elemento ***KeyValue*** contiene el valor de la clave que hay que aplicar a la firma.

El elemento que se encuentra dentro del *KeyValue* puede ser de los siguientes tipos definidos en el esquema xmldsig:

- DSAKeyValuue
- RSAKeyValue
- X509Data
- PGPData
- SPKIDataa
- MgmtData

La validación de las firmas de un documento XML Signature se lleva a cabo en dos pasos:

1. Se calcula la firma del elemento SignedInfo usando la información contenida el elemento KeyInfo (si existiera), y se compara con el valor que se encuentra en el elemento SignatureValue.
2. Si la primera validación se ha llevado a cabo de manera correcta, se recalcularán los DigestValue de cada objeto Reference encontrado en el elemento SignedInfo.

El trabajo con XML Signature es posible en Java a través de la JSR 105 (Java Specification Request #105), aprobada el día 6 de Junio de 2005. Esta especificación forma parte del WSDP de Java. La fundación Apache también dispone de implementación de una API para la generación y validación de XML Signature.

5.2.3.2 XML ENCRYPTION

XML Encryption es un estándar desarrollado por el W3C que describe el modo de utilizar XML para representar recursos de forma digital y codificada. La especificación está pensada para ser usada junto con la especificación de firmas digitales XML Signature para poder firmar y cifrar el contenido de los documentos XML.

Dispone de posibilidad de uso de varios algoritmos de codificación dejando así las puertas abiertas a la introducción de nuevos algoritmos que aparezcan en un futuro. Otra característica de XML Encryption es el soporte para diferentes granularidades en el cifrado de un documento.

Consideremos el documento XML (extraído de la especificación de sintaxis y procesado de XML Encryption) mostrado en la Figura 8, muestra un fragmento de información para un sistema de pago de un comercio. Contiene información relativa a una tarjeta de crédito: nombre, número, límite de operación, banco emisor y fecha de caducidad. Esta información es muy sensible como para circular por la red en texto plano.

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

Ilustración 7 . Fragmento XML sin cifrar

Podemos hacer uso de XML Encryption para cifrar partes del documento, siguiendo diferentes estrategias:

- Se puede cifrar un elemento completo, por ejemplo, en la Figura 8 se cifra el elemento ***CreditCard***, lo que hace que una simple inspección del documento cifrado ni siquiera revele la existencia de información de tarjeta de crédito.

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

Ilustración 8 Cifrado de un elemento completo

El elemento ***CypherData*** contiene la serialización del elemento CreditCard cifrado.

En algunos casos puede ser conveniente que un individuo conociera que el pago se está realizando mediante una tarjeta de crédito, con un límite de 5.000 USD, pero no publicar ni el número, ni la fecha de caducidad ni el emisor. En este caso, como muestra la Figura 8, se recurre al cifrado del contenido del elemento CreditCard, pero no se cifra el elemento CreditCard.

Otro posible escenario sería la situación en la que solamente fuera necesario cifrar el número de tarjeta de crédito, pero sin cifrar el propio elemento ***Number***. En este caso estaríamos cifrando el contenido de un elemento que solo contiene información de caracteres, no contiene otros elementos.

```
<?xml version='1.0'?>
<PaymentInfo xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>
      <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
       Type='http://www.w3.org/2001/04/xmlenc#Content'>
        <CipherData>
          <CipherValue>A23B45C56</CipherValue>
        </CipherData>
      </EncryptedData>
    </Number>
    <Issuer>Example Bank</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>
```

Ilustración 9 Cifrado del contenido de un elemento que contiene más elementos

Otras alternativas de granularidad para XML Encryption incluyen el cifrado de un documento completo, evitando cualquier detalle del mismo al no existir ningún elemento salvo los propios del cifrado. También es posible cifrar documentos cifrados (lo que se conoce como Súper-Encryption).

```
<?xml version='1.0'?>
<PaymentInfo xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>
      <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
       Type='http://www.w3.org/2001/04/xmlenc#Content'>
        <CipherData>
          <CipherValue>A23B45C56</CipherValue>
        </CipherData>
      </EncryptedData>
    </Number>
    <Issuer>Example Bank</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>
```

Ilustración 10 Cifrado del contenido de un elemento que contiene caracteres

XML Encryption soporta diferentes algoritmos de cifrado. Para el cifrado de bloques se citan los siguientes:

- TRIPLEDES
- AES128
- AES256
- AES192

No se especifican algoritmos para el cifrado de flujos de octetos, sin embargo si se hace referencia a otra serie de algoritmos para el intercambio de claves, resúmenes de los mensajes, y la autenticación de los mismos mediante XML Signature.

5.2.4 SEGURIDAD EN LA TRANSMISIÓN

Debido a que en Internet la información viaja a través de un medio completamente inseguro, es necesario establecer protocolos de comunicación que hagan dicho proceso de intercambio de información completamente seguro, esto se logra principalmente cifrando la información durante el tiempo en el que viaja de un extremo a otro del proceso de comunicación.

Los protocolos de seguridad utilizados en la actualidad son los siguientes:

- ✓ SSH (Secure Shell).- utilizado como reemplazo del comando telnet para establecer sesiones remotas.

- ✓ SSL (Secure Socket Layer).- utilizado en comunicaciones hipertexto principalmente, pero con aplicaciones en otros protocolos.

- ✓ TSL (Transport Layer Secure).- utilizado para establecer seguridad en la capa de transporte del modelo OSI.

- ✓ HTTPS (Hypertext Transfer Protocol Secure).- utilizado para establecer seguridad en el protocolo HTTP habitual.

- ✓ PGP (Pretty Good Privacy).- PGP es un criptosistema utilizado para garantizar proteger la información distribuida a través de Internet, mediante el uso de criptografía asimétrica, y facilitar la autenticación de documentos a través de las firmas digitales, aunque también puede ser utilizado para protección de datos almacenados en discos, copias de seguridad etc.

5.2.4.1 PROTOCOLO SSH

Es un protocolo para crear una conexión segura entre dos sistemas. En el protocolo SSH, la máquina cliente inicia una conexión con la maquina servidor, luego establece las siguientes medidas de seguridad:

- ✓ Después de una conexión inicial, el cliente verifica cada vez que se conecta al mismo servidor durante subsecuentes sesiones.

- ✓ El cliente transmite su información para autenticación al servidor, tal como su nombre de usuario y contraseña, pero cifrados ambos.
- ✓ Todos los datos enviados y recibidos durante la conexión, son transmitidos utilizando un cifrado de 128 bits.

Debido a que el protocolo SSH cifra todo lo que envía y recibe, por lo cual puede ser utilizado para asegurar diferentes protocolos inseguros, utilizando una técnica llamada port forwarding, un servidor SSH puede llegar a manejar protocolos inseguros como POP, etc.

5.2.4.1.1 COMPONENTES

SSH está compuesto por tres componentes:

- ✓ El protocolo de la capa de transporte [16] (SSH-TRANS), el cual provee autenticación del servidor, confidencialidad e integridad, y también opcionalmente compresión. La capa de transporte está corriendo sobre una

conexión TCP/IP generalmente, pero puede ser usada en cualquier otro flujo confiable de datos.

- ✓ El protocolo de autenticación de usuarios [17] (SSH-USERAUTH), autentica el usuario del lado del cliente con el servidor, se ejecuta sobre el protocolo de la capa de transporte.

- ✓ El protocolo de conexión (SSH-CONNECT) [18], el cual multiplexa el túnel cifrado en varios canales lógicos, se ejecuta sobre el protocolo de autenticación del usuario.

El cliente envía una petición de servicio una vez que una conexión de la capa segura de transporte ha sido establecida, una segunda petición de servicio es enviada después de que la autenticación del usuario ha sido completada, esto permite a los nuevos protocolos ser definidos y coexistir con los protocolos listados previamente.

El protocolo de conexión provee canales que pueden ser usados para un amplio rango de propósitos, los métodos estándar son desarrollados para abrir sesiones interactivas y para realizar un "túnel" para los puertos TCP/IP y las conexiones X11.

5.2.4.1.2 ARQUITECTURA

Cada servidor debería tener al menos una llave para cada uno de los algoritmos descritos en el FIPS-186-2[7] referentes a los estándares para firmas digitales. Dicha llave en el servidor, es utilizada durante el intercambio de llaves, para verificar que el cliente está realmente hablando con el servidor correcto, para que esto sea posible, el cliente debería tener un conocimiento previo de la llave publica del servidor host.

Este estándar define dos posibles modelos de confianza:

1. El cliente cuenta con una base de datos la cual asocia cada nombre de host, con su correspondiente llave publica, este modelo no requiere un

[7] http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf

mecanismo central de administración, y ningún tipo de coordinación por parte de una tercera persona, la desventaja de esto, es que la base de datos puede llegar a ser demasiado grande para mantener del lado del cliente.

2. El segundo esquema introduce la noción de una Autoridad Certificadora, que es una tercera persona la cual contiene las llaves de los diversos servidores, y únicamente el cliente conoce la llave de la propia autoridad certificadora, con esto, el cliente puede llegar a verificar la autenticidad de los servidores certificados por dicha autoridad.

El protocolo permite la opción de que el nombre del servidor y la llave del host no sean verificadas cuando se conecta al host por primera vez, permitiendo la comunicación entre ambas partes, esta conexión ofrece protección contra ataques pasivos, sin embargo es vulnerable a ataques de intruso en medio.

Las implementaciones actuales, permiten una conexión sin verificar únicamente la primera vez, la llave del servidor es almacenada localmente y comparada con la llave del host en todas las futuras conexiones.

5.2.4.1.3 PROPIEDADES DE SEGURIDAD

El objetivo principal del protocolo SSH, es mejorar la seguridad en el Internet, de acuerdo a las siguientes propiedades:

- ✓ Todos los algoritmos de cifrado, de integridad y de llave pública utilizados, son algoritmos bien conocidos, y de dominio público.

- ✓ Todos los algoritmos son utilizados con tamaños de llaves suficientemente grandes como para proveer seguridad contra ataques de criptoanálisis.

- ✓ Todos los algoritmos utilizados son periódicamente revisados, y en caso de que alguno haya sido vulnerado, es relativamente muy fácil reemplazarlo con otro que ofrezca el nivel de seguridad requerido

5.2.4.2 PROTOCOLO SSL

El objetivo principal del protocolo SSL [19], es garantizar privacidad y confiabilidad entre dos aplicaciones que se comunican; el protocolo está compuesto por dos capas, la de más bajo nivel se basa en alguno de los protocolos de transporte más confiables, se denomina SSL Record Protocol, el cual es utilizado para encapsular varios protocolos de nivel más alto, uno de esos protocolos es el SSL Handshake Protocol, el cual permite al servidor y al cliente, autenticarse uno al otro, y negociar el algoritmo de cifrado y las llaves criptográficas antes de que el protocolo transmita o reciba el primer byte de datos.

Una ventaja de SSL es que es independiente del protocolo, un protocolo de más alto nivel puede estar en una capa superior al protocolo SSL de forma transparente (la mayoría de las veces es TCP).

El protocolo SSL tiene tres propiedades básicas de seguridad:

- ✓ La conexión es privada.- Se realiza un proceso de autenticación inicial, para establecer una llave secreta, después se utiliza criptografía simétrica para el cifrado de datos.

- ✓ Las identidades tanto del cliente como del servidor, pueden ser autenticadas utilizando criptografía asimétrica o de llave pública.

- ✓ La conexión es confiable.- el transporte de los mensajes, incluye una revisión de la integridad del mensaje, utilizando una MAC[8] con llave, con algoritmos seguros (SHA).

Los objetivos del protocolo SSL v3.0 (la última versión), en orden de prioridad, consisten en:

- ✓ Seguridad criptográfica.- SSL es utilizado para establecer una conexión segura entre dos partes.

[8] Message Authentication Code, es una pieza de información utilizada para autenticar un mensaje. Un algoritmo MAC acepta como entrada una llave secreta y el mensaje, desarrolla algunas funciones matemáticas y arroja como salida un MAC, el cual garantiza tanto la integridad del mensaje como la autenticidad.

- ✓ Interoperabilidad.- Los programadores deben ser capaces de poder desarrollar aplicaciones utilizando SSL 3.0 y poder intercambiar parámetros criptográficos, sin el conocimiento del código de los demás.

- ✓ Extensibilidad.- SSL busca ofrecer un marco de trabajo en el cual, nuevas llaves públicas y métodos de cifrado, puedan ser incorporados como sea necesario; esto con el objeto de prevenir la necesidad de crear un nuevo protocolo (con el riesgo de la introducción de posibles nuevas debilidades), y evitar la necesidad de tener que implementar una nueva librería de seguridad completa.

- ✓ Eficiencia relativa.- Las operaciones criptográficas tienden a consumir muchos recursos, principalmente tiempo de procesamiento, particularmente las que involucran criptografía asimétrica, por esta razón, el protocolo SSL ha incorporado un esquema opcional de manejo de sesión optimizado, para reducir el número de conexiones que necesitan ser establecidas, adicionalmente, se ha tenido cuidado en reducir el tráfico en la red.

5.2.4.2.1 DESCRIPCIÓN DEL PROTOCOLO

SSL es un protocolo dividido en capas, en cada una de ellas, los mensajes pueden incluir campos para la longitud, descripción y contenido. SSL toma los mensajes que van a ser transmitidos, fragmentos de datos en bloques manejables, opcionalmente comprime los datos, aplica MAC, cifra y transmite el resultado. Los datos recibidos son descifrados, verificados, descomprimidos y ensamblados, entonces entregados a los clientes de más alto nivel.

5.2.4.3 PROTOCOLO TLS (TRANSPORT LAYER SECURITY)

El protocolo TLS[9] [7], es un protocolo para establecer una conexión segura entre un cliente y un servidor, TLS es capaz de autenticar en ambos lados de la comunicación, y crea una conexión cifrada entre las dos. El protocolo TLS puede ser extendido, esto es que nuevos algoritmos pueden ser utilizados para cualquiera de los propósitos, con la condición de que tanto el cliente como el servidor estén conscientes de los algoritmos.

[9] definido en el RFC 2246,

La principal propiedad del protocolo TLS, es ofrecer privacidad e integridad de los datos, entre dos aplicaciones que se comunican; el protocolo está compuesto por dos capas, el TLS Record Protocol y el TLS Handshake Protocol.

El TLS Record Protocol ofrece seguridad en las conexiones y tiene dos propiedades básicas:

1. La conexión es privada; se utiliza criptografía simétrica para el cifrado de los datos (DES, AES, RC4, entre otros), las llaves para los algoritmos simétricos son generadas una sola vez para cada sesión, y están basadas en un secreto negociado por otro protocolo (TLS Handshake), el TLS Record Protocol puede ser utilizado sin cifrado también.

2. La conexión es confiable, el transporte de mensajes, incluye una verificación de integridad de mensajes, utilizando una MAC con llave, para esto se utilizan algoritmos de funciones de hash seguros como SHA1, SHA256, MD5.

El TLS Record Protocol, es utilizado para la encapsulación de varios protocolos de nivel superior, uno de tales protocolos encapsulados, es el TLS Handshake Protocol, el cual es utilizado para autenticar tanto a los clientes como a los servidores, y para negociar un algoritmo de cifrado así como las llaves criptográficas, antes de que el protocolo de la aplicación transmita o reciba el primer byte de datos. El protocolo TLS Handshake Protocol ofrece seguridad en la conexión, y tiene 3 propiedades básicas:

1. La identidad de la otra parte puede ser verificada utilizando criptografía asimétrica (RSA o DSS), esta autenticación puede ser opcional, pero generalmente se requiere por al menos una de las partes.

2. La negociación de un secreto compartido es segura: el secreto negociado no está disponible para ningún atacante, incluso si el atacante utilizara un ataque hombre en el medio.

3. La negociación es confiable: ningún atacante puede modificar la comunicación sin ser detectado por las partes que intervienen en la comunicación.

Una ventaja de TLS, es que es independiente del protocolo de la aplicación, los protocolos de más alto nivel pueden tener capas encima del protocolo TLS de forma transparente, sin embargo, el estándar TLS no especifica de qué forma los protocolos definen la seguridad en TLS, las decisiones de cómo inicializar ***handshaking***, y cómo interpretar los certificados de autenticación intercambiados, se dejan a juicio de los diseñadores y los implementadores de los protocolos que corren por encima de TLS.

5.2.4.4 PROTOCOLO HTTPS

El protocolo HTTPS es una versión segura del protocolo http, utiliza un sistema de cifrado basado en la Secure Socket Layers (SSL), para crear un canal seguro cuyo nivel de cifrado depende del servidor remoto y del navegador utilizado por el cliente, más apropiado para el tráfico de información sensible que el protocolo http, ambos protocolos pueden existir juntos, ya que los navegadores de hoy en día, los soportan.

El protocolo http[10] , fue originalmente utilizado en claro sobre Internet, sin embargo, el uso reiterado de dicho protocolo en aplicaciones importantes, ha requerido que algunas medidas de seguridad sean consideradas, por lo cual los protocolos SSL y su sucesor TLS (ambos descritos previamente), fueran diseñados para ofrecer seguridad en el canal de comunicación.

Conceptualmente hablando, el protocolo http puede utilizarse sobre algún otro protocolo de seguridad, TLS o SSL, de la misma forma en que se utiliza sobre el protocolo TCP.

5.2.4.4.1 INICIO DE UNA CONEXIÓN

El agente que actúa como el cliente http, también debería actuar como el cliente TLS o SSL, y sería el encargado de iniciar una conexión al servidor en el puerto apropiado, y enviar un mensaje de inicio del protocolo, por ejemplo un mensaje del tipo TLS ClientHello para iniciar el protocolo de acuerdo (Handshake) sobre TLS. Una vez que el protocolo de acuerdo (Handshake) ha terminado, el cliente debe

[10] La definición del protocolo HTTP reencuentra en el RFC 2616, documento que puede ser localizado en la dirección: http://www.faqs.org/rfcs/rfc2616.html

entonces iniciar la primera petición http; todos los datos http deben ser enviados como datos de la aplicación TLS.

5.2.4.4.2 FIN DE UNA CONEXIÓN

EL protocolo TLS ofrece una posibilidad de cerrar una conexión de forma segura; cuando se recibe un mensaje de petición de cierre de la conexión, una implementación debe asegurar que ya no se recibirán datos después de haber cerrado dicha conexión.

Las implementaciones TLS deben iniciar un intercambio de mensajes de cierre, antes de cerrar por completo una conexión, sin embargo, una implementación TLS puede después de enviar una advertencia de cierre, proceder a cerrar la conexión sin tener que esperar a que la otra parte en el canal de comunicación envíe su advertencia de cierre de conexión, generando así un cierre incompleto.

Cuando HTTPS fue utilizado por primera vez por el navegador Netscape 2 en 1995, la estrategia utilizada fue la de usar dos puertos diferentes, el puerto 80 para http y el 443 para HTTPS respectivamente.

Dos posibles alternativas para proveer seguridad en Web Services:

- Seguridad en Capa de Transporte (SSL/TLS)
- Seguridad a nivel de Mensaje SOAP

Se provee seguridad únicamente durante la transmisión de los mensajes. Si los intermediarios no son seguros, no es posible garantizar seguridad en todo el trayecto del mensaje. Las características de seguridad se aplican a todo el mensaje, por lo que no es posible cifrar sólo partes de los mensajes transmitidos. Se depende en gran medida del protocolo de transporte utilizado para proveer las características de seguridad.

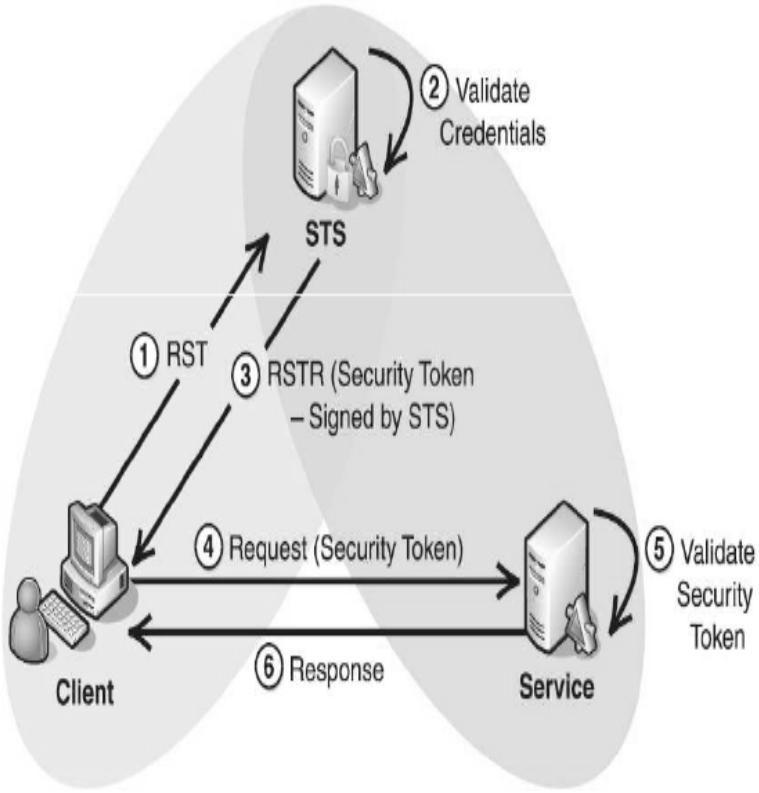


Ilustración 11 Protocolo de transporte

6 MECANISMOS DE SEGURIDAD DE LA INFORMACIÓN EN APLICACIONES WEB

Para el sistema financiero es de gran importancia que su información sea de carácter confidencial, por tanto se requiere de mecanismos de seguridad que garanticen que dicha información no será modificada, sustraída o falsificada por personas ajenas. Esto es un sistema Web debe garantizar la autenticidad, integridad y confidencialidad de toda la información involucrada en el sistema.

La *autenticidad* consiste en que la información provenga de quien realmente dice provenir, esto se logra mediante la implementación de firmas digitales principalmente; la integridad consiste en que la información no haya sido modificada en ningún aspecto, que permanezca tal cual originalmente, mientras que la confidencialidad consiste en que la información no pueda ser sustraída por terceras personas.

Una aplicación Web que se diga segura, debe contemplar mecanismos que garanticen que efectivamente ninguna persona ajena al sistema, puede modificar, obtener o falsificar datos de dicha aplicación. En la medida en que las aplicaciones Web críticas (que involucren información confidencial como números de tarjetas de crédito, números de identificación personal, etc.) se consideren seguras, los usuarios harán un mayor uso de ellas, sin el temor de que la información que están proporcionando va a ser utilizada de forma ilícita.

6.1 *CRIPTOGRAFÍA*

La criptografía es una ciencia que utiliza las matemáticas para cifrar y descifrar datos, la criptografía permite almacenar información sensible, o transmitirla a través de canales inseguros (como Internet) de tal forma que no pueda ser leída por nadie que no sea la persona a la que va dirigida, o que tiene los permisos para hacerlo.

El criptoanálisis es la ciencia encargada de analizar, y romper la comunicación segura, las técnicas clásicas de criptoanálisis involucran una combinación de

razonamiento analítico, la aplicación de herramientas matemáticas, búsqueda de patrones, paciencia, determinación y un poco de suerte.

La Criptología involucra tanto a la Criptografía como al Criptoanálisis, mientras que un criptosistema está formado por un algoritmo criptográfico, más todas las posibles llaves y los protocolos.

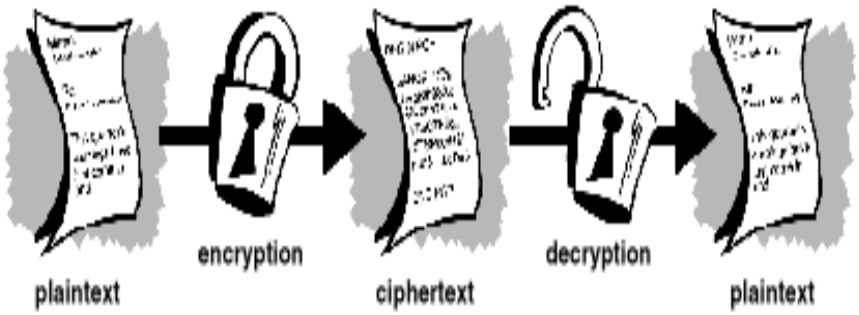


Ilustración 12 Criptografía

"*Existen dos tipos de criptografía en este mundo, la criptografía que evitará que tu hermano pequeño pueda leer tus archivos confidenciales, y la criptografía que evitará que los gobiernos más poderosos del mundo puedan leer tus archivos confidenciales***" [10].**

La fortaleza de una aplicación criptográfica es medida de acuerdo al tiempo y a los recursos necesarios para a partir de un texto cifrado, recuperar el contenido del texto original sin tener conocimiento de la llave, esto es utilizando algún tipo de ataque. El resultado de una criptografía fuerte o robusta, es que ésta es muy difícil de descifrar sin la utilización de la herramienta de decodificación apropiada.

Un ***algoritmo criptográfico*** o ***cifrador*****,** es una función matemática utilizada en el proceso de cifrar y descifrar. Un algoritmo criptográfico trabaja en combinación con una llave secreta, que puede ser una palabra, número o frase, la cual será utilizada para el cifrado del texto plano.

Es necesario recalcar, que el texto cifrado resultante va a ser diferente cuando las llaves de cifrado son diferentes, por lo que la seguridad de los datos cifrados

depende completamente de dos cosas: de la robustez del algoritmo de cifrado y la secrecía de la llave.

6.1.1 LLAVES

Una llave es un valor que trabaja con un algoritmo criptográfico para producir un texto cifrado específico; las llaves son básicamente números muy grandes, y su tamaño se mide en bits, y entre más grande es la llave, más seguro es el texto cifrado.

Mientras que las llaves públicas y privadas de la criptografía asimétrica están completamente relacionadas, es muy difícil deducir la llave privada a partir de la llave pública, sin embargo, si es posible hacerlo teniendo suficiente tiempo y poder de cómputo, por lo cual es necesario tener llaves del tamaño correcto, esto es suficientemente grandes para ser seguras, pero suficientemente pequeñas para que para ser aplicadas con cierta facilidad.

Las llaves más largas, serán criptográficamente seguras por un periodo de tiempo más largo, éstas deben ser almacenadas en forma cifrada también, para evitar que algún intruso pueda tener acceso a nuestro disco duro y obtener la llave.

6.1.2 FUNCIONES HASH DE UN SENTIDO

Una función ***Hash,*** toma una entrada de longitud variable, un mensaje de cualquier tamaño, incluso de algunos miles de millones de bits, y produce una salida de longitud fija, con la condición de que si alguno de los bits del mensaje original es modificado, la salida producida por la función de hash, va a ser completamente diferente. A esta salida se le conoce como la ***firma*** o ***resumen del mensaje***, (message digest). Actualmente los algoritmos hash más utilizados son el MD5 (sucesor del algoritmo MD4) y SHA **[11]** en sus versiones SHA-1 **[12],** SHA-224, SHA-256 y SHA-512, no obstante se han documentado algunas vulnerabilidades en algunos de ellos1[11].

[11] Las vulnerabilidades documentadas se refieren a ataques de colisión sobre los algoritmos de hash, para mayor información sobre estos ataques, consultar la referencia:
http://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html

Las funciones de hash tienen dos propiedades básicas, la primera es que son de un solo sentido, esto significa que es relativamente muy fácil tomar un mensaje y calcular su valor de hash, pero es prácticamente imposible tomar un valor de hash y obtener a partir de él, el valor del mensaje original, (entendiendo por imposible que no puede ser hecho en una cantidad razonable de tiempo); la segunda propiedad, es que los algoritmos de hash son libres de colisiones, esto significa que es imposible encontrar dos mensajes que tengan el mismo valor del hash. Vulnerar un algoritmo de hash, implica que alguna o ambas no se cumpla.

La propiedad de libertad de colisiones es parcialmente cierta, ya que como hemos mencionado anteriormente, cada algoritmo de hash produce una salida fija en tamaño, tomando por ejemplo los 160 bits que arroja SHA-1, tenemos que el espacio completo de posibilidades de valores hash es 2^{160}, un número bastante grande pero evidentemente menor al infinito de posibilidades de mensajes que podemos utilizar, por lo cual existe un número infinito de colisiones, para el caso de SHA-1, obteniendo 2^{80} mensajes aleatorios, llegaríamos a una colisión de forma segura, eso sería un ataque de fuerza bruta sobre el algoritmo.

6.2 *CRIPTOGRAFÍA SIMÉTRICA*

La criptografía simétrica o convencional, utiliza únicamente una sola llave, la cual es utilizada tanto para cifrar como para descifrar, **DES (Data Encryption Standard),** es un ejemplo de un criptosistema simétrico que utiliza llaves de 56 bits, definido como estándar por la Secretaría de Comercio de los Estados Unidos el 22 de enero de 1988 **[13]**, y posteriormente reemplazado por **AES (Advanced Encryption Standard)**, publicado en el FIPS PUB 197 de fecha 26 de noviembre del 2001 **[14]**, el cual utiliza llaves de 128,192 ó 256 bits.

El cifrado convencional (criptografía simétrica) tiene como principal ventaja, que es muy rápido de ejecutar, es muy útil cuando se tienen que cifrar una gran cantidad de datos, sin embargo tiene la desventaja que se tiene que transmitir la llave por un canal seguro.

Cuando un emisor y un receptor se tienen que comunicar de forma segura utilizando criptografía simétrica, entonces ellos tienen que quedar de acuerdo en la llave que han de utilizar, y mantenerla en secreto, pero si ellos se encuentran en diferentes lugares, entonces ellos necesitan un mecanismo seguro para poder intercambiar la llave secreta, ya que cualquier persona que pueda interceptar el

mensaje que contiene la llave, podrá descifrar toda la comunicación entre las dos personas.

La ***figura 13***, ilustra el esquema de cifrado utilizando criptografía simétrica, teniendo una sola llave tanto para cifrar como para descifrar.

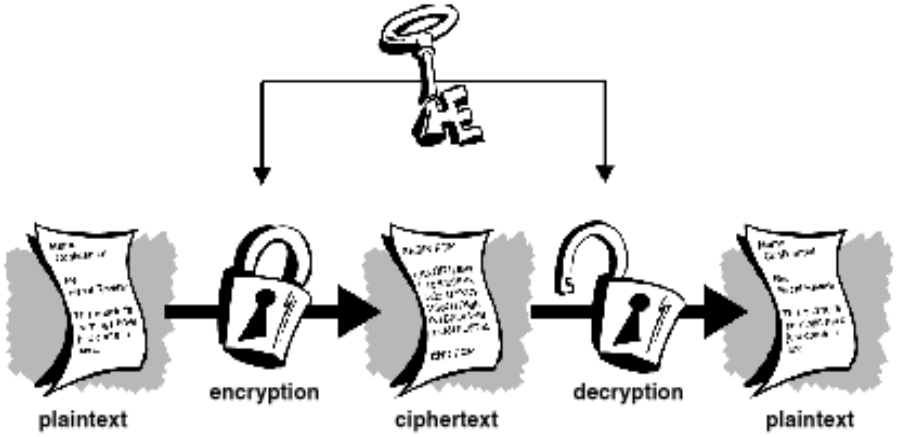


Ilustración 13 Criptografía simétrica

6.3 CRIPTOGRAFÍA ASIMÉTRICA

El concepto de criptografía asimétrica fue introducido por *Whitfield Diffie y Martin Hellman* en 1975, aunque hay evidencia de que el servicio secreto británico lo había inventado algunos años antes, solo que lo mantienen como secreto militar. La criptografía asimétrica utiliza un par de llaves, una llave que puede ser pública, utilizada generalmente para cifrar, y la correspondiente llave privada, utilizada para descifrar la información cifrada con la pública.

Si A desea enviar un mensaje cifrado a B, entonces A obtiene la llave pública de B y cifra el mensaje con ella, únicamente B con su llave privada puede descifrar lo que A le escribió. Computacionalmente hablando, es imposible deducir la llave privada a partir de la pública, esto implica que cualquiera que tenga la llave pública puede únicamente cifrar, no descifrar el texto.

El beneficio principal de la criptografía asimétrica, es que permite que personas que no tienen un acuerdo de seguridad previo, puedan intercambiar información de forma segura, y por lo tanto la necesidad de enviar y recibir las llaves secretas

únicamente a través de algún canal seguro es eliminada, puesto que todas las comunicaciones involucran únicamente llaves públicas, y ninguna llave privada es transmitida o compartida. Algunos ejemplos de criptosistemas de llave pública son ***Elgamal, RSA, Diffie-Hellman*** *y* ***DSA (Digital Signature Algorithm).***

No obstante las ventajas ofrecidas por la criptografía simétrica, existen algunas desventajas descritas por el criptólogo Bruce Schneier **[15]**, entre las cuales podemos mencionar:

- ¿Qué tan segura está nuestra llave privada?- regularmente las llaves privadas son almacenadas de forma cifrada en el disco duro de nuestras propias computadoras o en dispositivos extraíbles.

- ¿De qué forma la Autoridad Certificadora verifica las identidades de sus clientes?, nosotros podemos obtener el Certificado Digital de alguna persona, pero ¿cómo podemos saber que esa persona es realmente la que pensamos que es?

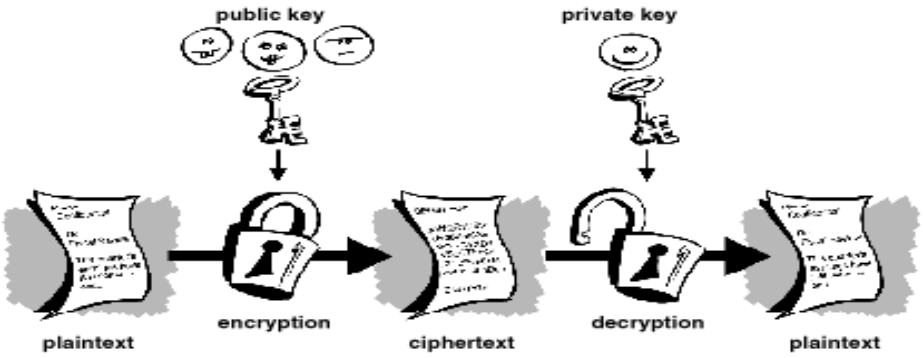


Ilustración 14 Criptografía Asimétrica

6.4 DES (ESTÁNDAR DE CIFRADO DE DATOS), DESCIFRADO DE LA CLAVE SECRETA

El 15 de mayo de 1973, el **NBS** (*National Bureau of Standars, en castellano: Agencia Nacional de Normalización*) hoy en día denominada *NIST* (*National Institute of Standars and Technology, en castellano: Instituto Nacional de*

Normalización y Tecnología), hizo un llamamiento en el *Federal Register (el equivalente en España del Boletín Oficial del Estado)* para la creación de un algoritmo de cifrado que cumpliera con los siguientes requisitos:

- ofrecer un alto nivel de seguridad relacionado con una pequeña clave utilizada para cifrado y descifrado.
- ser comprensible
- no depender de la confidencialidad del algoritmo
- ser adaptable y económico
- ser eficaz y exportable

A finales de 1974, IBM propuso "Lucifer", que gracias a la NSA (National Standard Agency, en castellano: Agencia Nacional de Seguridad) fue modificado el 23 de noviembre de 1976, convirtiéndose en **DES** (*Data Encryption Standard, en castellano: Estándar de Cifrado de Datos*). El DES fue aprobado por el NBS en 1978. El DES fue estandarizado por el *ANSI* (*American National Standard Institute, en castellano: Instituto Nacional Americano de Normalización*) bajo el nombre de *ANSI X3.92*, más conocido como *DEA* (*Data Encryption Algorithm, en castellano: Algoritmo de Cifrado de Datos*).

6.4.1 ALGORITMOS PARA CLAVES PRIVADAS

6.4.1.1 TRIPLE DES (3DES)

El 15 de mayo de 1973, el **NBS** (*National Bureau of Standars, en castellano: Agencia Nacional de Normalización*) hoy en día denominada *NIST* (*National Institute of Standars and Technology, en castellano: Instituto Nacional de Normalización y Tecnología*), hizo un llamamiento en el *Federal Register (el equivalente en España del Boletín Oficial del Estado)* para la creación de un algoritmo de cifrado que cumpliera con los siguientes requisitos:

- ofrecer un alto nivel de seguridad relacionado con una pequeña clave utilizada para cifrado y descifrado

- ser comprensible
- no depender de la confidencialidad del algoritmo
- ser adaptable y económico
- ser eficaz y exportable

A finales de 1974, IBM propuso "Lucifer", que gracias a la NSA (National Standard Agency, en castellano: Agencia Nacional de Seguridad) fue modificado el 23 de noviembre de 1976, convirtiéndose en **DES** (*Data Encryption Standard, en castellano: Estándar de Cifrado de Datos*). El DES fue aprobado por el NBS en 1978. El DES fue estandarizado por el *ANSI* (*American National Standard Institute, en castellano: Instituto Nacional Americano de Normalización*) bajo el nombre de *ANSI X3.92*, más conocido como *DEA* (*Data Encryption Algorithm, en castellano: Algoritmo de Cifrado de Datos*).

Triple DES trata de repetir el algoritmo DES tres veces en el texto plano, usando dos o tres llaves diferentes para producir el texto Cifrado. Este es un algoritmo usado en el criptoanálisis por ser muy resistente actuando como una alternativa para agregar seguridad contra la vulnerabilidad potencial de DES contra ataques.

Se trata de un sistema de cifrado simétrico por bloques de 64 bits, de los que 8 bits (un byte) se utilizan como control de paridad (para la verificación de la integridad de la clave). Cada uno de los bits de la clave de paridad (1 cada 8 bits) se utiliza para controlar uno de los bytes de la clave por paridad impar, es decir, que cada uno de los bits de paridad se ajusta para que tenga un número impar de "1" dentro del byte al que pertenece. Por lo tanto, la clave tiene una longitud "útil" de 56 bits, es decir, realmente sólo se utilizan 56 bits en el algoritmo.

El algoritmo se encarga de realizar combinaciones, sustituciones y permutaciones entre el texto a cifrar y la clave, asegurándose al mismo tiempo de que las operaciones puedan realizarse en ambas direcciones (para el descifrado). La combinación entre sustituciones y permutaciones se llama **cifrado del producto**.

La clave es codificada en 64 bits y se compone de 16 bloques de 4 bits, generalmente anotados de k_1 a k_{16}. Dado que "solamente" 56 bits sirven para el cifrado, puede haber hasta 2^{56} (o $7.2*10^{16}$) claves diferentes.

El algoritmo DES

Las partes principales del algoritmo son las siguientes:

- fraccionamiento del texto en bloques de 64 bits (8 bytes)
- permutación inicial de los bloques
- partición de los bloques en dos partes: izquierda y derecha denominadas *I* y *D* respectivamente
- fases de permutación y de sustitución repetidas 16 veces (denominadas **rondas**)
- reconexión de las partes izquierda y derecha, seguida de la permutación inicial inversa.

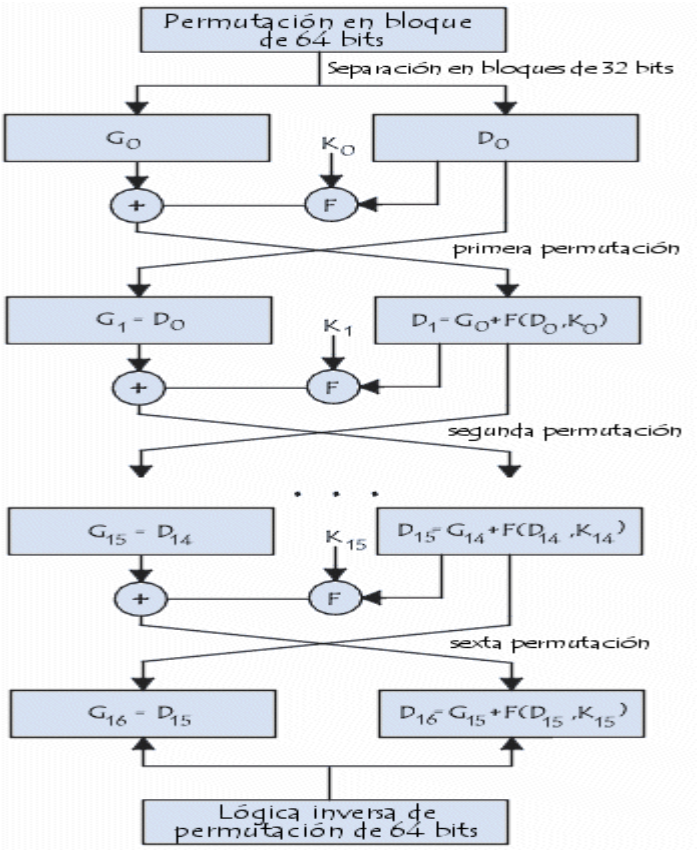


Ilustración 15 Algoritmo DES

6.5 FIRMAS DIGITALES

Las firmas digitales son un mecanismo que permite el receptor de un mensaje, verificar la autenticidad del origen de la información, así como verificar la integridad de la información recibida. Una firma digital ofrece así mismo, el esquema de “no repudio”, lo que significa que una persona que ha firmado un documento, no puede negar el hecho de haberlo firmado.

La forma de crear las firmas digitales es la siguiente: primero se calcula el digest o resumen del mensaje que se desea firmar, ese resumen es cifrado con la llave privada de la persona que firma el mensaje, y dicha firma es adjuntada al mensaje original.

Para verificar la firma digital contenida en un documento, el procedimiento es el siguiente: el receptor que desea verificar la firma digital del emisor descifra la firma contenida en el mensaje utilizando la llave pública del emisor, después vuelve a calcular el resumen (hash) del documento recibido y compara estos dos valores, si coinciden, se puede garantizar que el emisor firmó digitalmente el documento, , y que el documento no fue modificado durante su recorrido desde el emisor hacia el receptor.

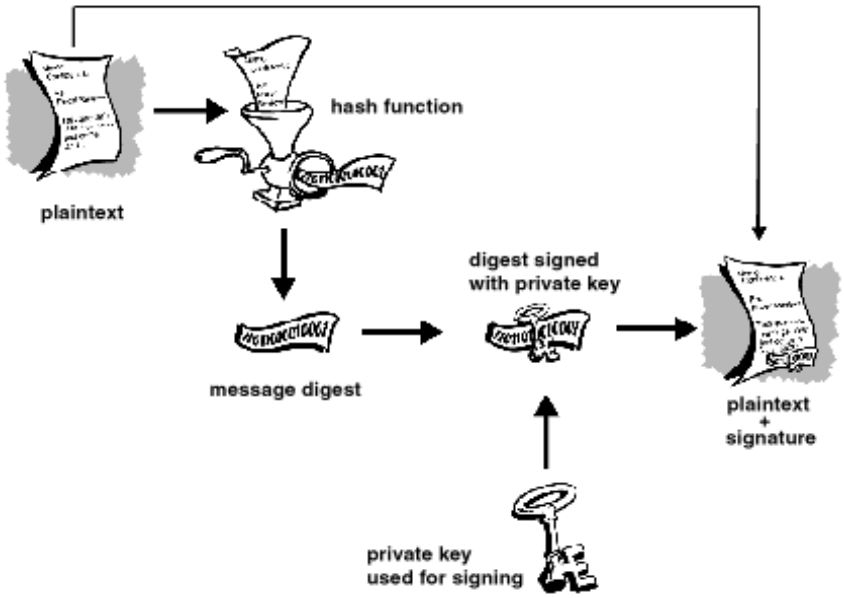


Ilustración 16 Procedimiento para firmar digitalmente un documento

6.6 *CERTIFICADOS DIGITALES*

Un problema que tienen los criptosistemas de llave pública, es que los usuarios deben de estar completamente seguros de que están cifrando información con la llave pública de la persona correcta, en un ambiente en donde es relativamente muy fácil intercambiar llaves por medio de servidores públicos, es muy probable que los ataques de *-hombre en el medio-* se presenten, es decir que cuando algún atacante suplanta la llave de la persona a la que se le quiere enviar un mensaje cifrado, con otra llave con la cual él conoce la respectiva llave privada.

Toda la información dirigida hacia el verdadero receptor es entonces interceptada por el atacante, el cual al conocer la verdadera llave pública del receptor, reenvía el mensaje cifrándolo con su verdadera llave pública, actuando el atacante como un punto intermedio en la comunicación entre emisor y receptor.

En un ambiente de llave pública, a veces es necesario intercambiar información con gente que nunca se ha conocido, es entonces cuando surge la necesidad de utilizar los certificados digitales, que se pueden definir como una credencial de identidad, la cual contiene información certificada acerca de la persona propietaria, contiene una llave pública y la(s) firma(s) digital(es) de la(s) Autoridad(es) Certificadora(s) que emitió o emitieron dicho Certificado Digital.

El formato estándar para los Certificados Digitales es el X.509, el cual incluye como información:

1. El número de versión X.509 del Certificado Digital.
2. La llave pública del dueño del Certificado Digital.
3. El número de serie del Certificado Digital.
4. El identificador único del dueño del Certificado Digital.
5. El periodo de validez del Certificado Digital.
6. El nombre del emisor del Certificado Digital.
7. La firma digital del emisor del Certificado Digital, utilizando su llave privada.
8. El identificador del algoritmo de la firma digital.

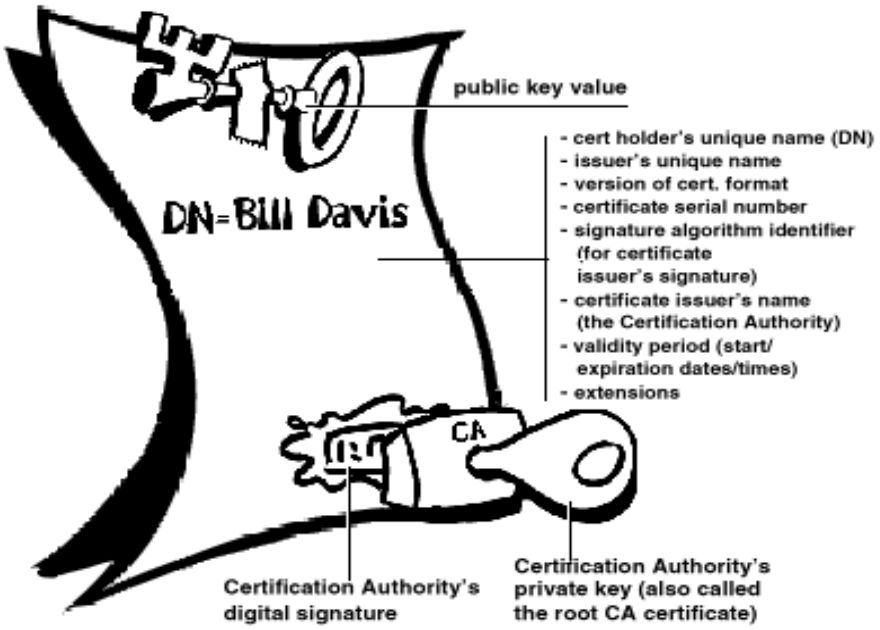


Ilustración 17 Representación de un Certifico Digital X.509

6.6.1 CERTIFICADO X.509

El formato de certificados X.509 es un estándar del ITU-T (*International Telecommunication Union-Telecommunication Standarization Sector*) y el ISO/IEC (*International Standards Organization / International Electrotechnical Commission*) que se publicó por primera vez en 1988. El formato de la versión 1 fue extendido en 1993 para incluir dos nuevos campos que permiten soportar el control de acceso a directorios. Después de emplear el X.509 v2 para intentar desarrollar un estándar de correo electrónico seguro, el formato fue revisado para permitir la extensión con campos adicionales, dando lugar al X.509 v3, publicado en 1996.

Los elementos del formato de un certificado X.509 v3 son:

- ✓ **Versión**. El campo de versión contiene el número de versión del certificado codificado. Los valores aceptables son 1, 2 y 3.

- ✓ **Número de serie del certificado**. Este campo es un entero asignado por la autoridad certificadora. Cada certificado emitido por una CA debe tener un número de serie único.

- ✓ **Identificador del algoritmo de firmado**. Este campo identifica el algoritmo empleado para firmar el certificado (como por ejemplo el RSA o el DSA).

- ✓ **Nombre del emisor**. Este campo identifica la CA que ha firmado y emitido el certificado.

- ✓ **Periodo de validez**. Este campo indica el periodo de tiempo durante el cual el certificado es válido y la CA está obligada a mantener información sobre el estado del mismo. El campo consiste en una fecha inicial, la fecha en la que el certificado empieza a ser válido y la fecha después de la cual el certificado deja de serlo.

- ✓ **Nombre del sujeto**. Este campo identifica la identidad cuya clave pública está certificada en el campo siguiente. El nombre debe ser único para cada entidad certificada por una CA dada, aunque puede emitir más de un certificado con el mismo nombre si es para la misma entidad.

- ✓ **Información de clave pública del sujeto**. Este campo contiene la clave pública, sus parámetros y el identificador del algoritmo con el que se emplea la clave.

- ✓ **Identificador único del emisor**. Este es un campo opcional que permite reutilizar nombres de emisor.

- ✓ **Identificador único del sujeto**. Este es un campo opcional que permite reutilizar nombres de sujeto.

- ✓ **Extensiones**. Las extensiones del X.509 v3 proporcionan una manera de asociar información adicional a sujetos, claves públicas, etc. Un campo de extensión tiene tres partes:

    - **Tipo de extensión**. Es un identificador de objeto que proporciona la semántica y el tipo de información (cadena de texto, fecha u otra estructura de datos) para un valor de extensión.

    - **Valor de la extensión**. Este subcampo contiene el valor actual del campo.

    - **Indicador de importancia**. Es un *flag* que indica a una aplicación si es seguro ignorar el campo de extensión si no reconoce el tipo. El indicador proporciona una manera de implementar aplicaciones que trabajan de modo seguro con certificados y evolucionan conforme se van añadiendo nuevas extensiones.

El ITU y el ISO/IEC han desarrollado y publicado un conjunto de extensiones estándar en el apéndice al X.509 v3:

- **Limitaciones básicas**. Este campo indica si el sujeto del certificado es una CA y el máximo nivel de profundidad de un camino de certificación a través de esa CA.

- **Política de certificación**. Este campo contiene las condiciones bajo las que la CA emitió el certificado y el propósito del certificado.

- **Uso de la clave**. Este campo restringe el propósito de la clave pública certificada, indicando, por ejemplo, que la clave sólo se debe usar para firmar, para la encriptación de claves, para la encriptación de datos, etc. Este campo suele marcarse como importante, ya que la clave sólo está certificada para un propósito y usarla para otro no estaría validado en el certificado.

El formato de certificados X.509 se especifica en un sistema de notación denominado *sintaxis abstracta uno* (*Abstract Sintax One* o ASN-1). Para la transmisión de los datos se aplica el DER (***Distinguished Encoding Rules*** o *reglas de codificación distinguible*), que transforma el certificado en formato ASN-1 en una secuencia de octetos apropiada para la transmisión en redes reales.

6.7 *PGP (PRETTY GOOD PRIVACY)*

Es un sistema desarrollado por *Phil Zimmerman* cuya finalidad es proteger la información distribuida a través de Internet mediante el uso de criptografía de asimétrica y firmas digitales, también puede ser utilizado PGP para proteger datos almacenados en discos. PGP combina algunas de las mejores características de los sistemas criptográficos simétricos y asimétricos, por lo cual PGP es un criptosistema híbrido.

Cuando un usuario cifra un texto plano con PGP, el texto es comprimido primeramente, esto con la finalidad de ahorrar tiempo de transmisión y espacio en disco, pero sobretodo, la compresión del texto incrementa la seguridad criptográfica, ya que la mayoría de las técnicas de criptoanálisis explotan patrones encontrados en el texto plano para vulnerar el cifrador (criptoanálisis de frecuencias).

6.7.1 CIFRANDO CON PGP

PGP comienza creando una llave de sesión, la cual es una llave secreta que se utiliza una sola vez, esta llave es un número aleatorio generado utilizando como función generadora, los movimientos del ratón, esta llave de sesión trabaja con un algoritmo de cifrado simétrico seguro y muy rápido, para cifrar el texto plano, el resultado es el texto cifrado. Una vez que los datos son cifrados, la llave de sesión es entonces cifrada con la llave pública del que recibe el mensaje, junto con esta llave cifrada, se envía también el texto cifrado.

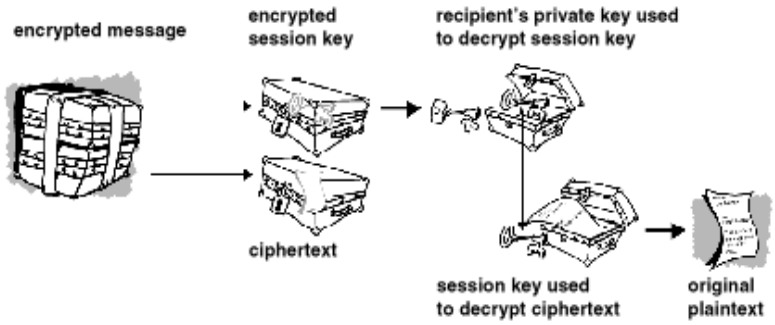


Ilustración 18 Proceso de cifrado utilizando PGP

6.7.2 DESCIFRANDO CON PGP

Para el proceso de descifrar, el receptor del mensaje utiliza su llave privada para descifrar la llave de sesión que viene acompañando al mensaje cifrado, entonces utilizando la llave de sesión, se descifra el texto cifrado. La combinación de los dos métodos de criptografía, combina la practicidad de la asimétrica con la velocidad de la criptografía simétrica, la cual es en proporción mil veces más rápida.

La criptografía de llave pública, ofrece una solución al problema de distribución de llaves y transmisión de datos, y utilizados ambos sistemas, se obtiene una solución mejorada, sin sacrificar en seguridad.

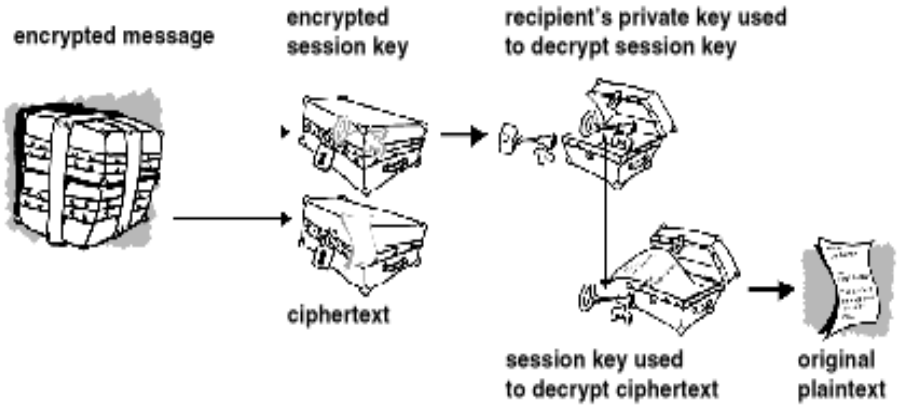


Ilustración 19 Proceso de descifrado con PGP

7 ARQUITECTURA ORIENTADA A SERVICIOS (SOA)

Las compañías líderes obtienen eficiencia operacional y agilidad de negocios mediante procesos y servicios de negocio adaptables, reutilizables, creados sobre la base de una Arquitectura Orientada a Servicios (SOA) verdaderamente manejables; partiendo de la organización de los sistemas, aplicaciones y datos que conforman las Tecnologías de Información (TI), de manera que se permita a los sistemas distribuidos heterogéneos y aplicaciones complejas transformarse en una red de recursos integrados, simplificados y sumamente flexibles.

Un proyecto SOA bien ejecutado permite alinear los recursos de las TI de forma más directa con los objetivos de negocio, ganando así un mayor grado de integración con clientes y proveedores, proporcionando una inteligencia de negocio más precisa y más accesible con la cual se podrán adoptar mejores decisiones, y ayuda a las empresas a optimizar sus procesos internos y sus flujos de información para mejorar la productividad individual. El resultado neto es un aumento notable de la agilidad de la organización.

Los servicios Web son la forma más habitual de implementar SOA. Los servicios Web son aplicaciones que utilizan estándares para el transporte, codificación y protocolo de intercambio de información. Los SW permiten la intercomunicación entre sistemas de cualquier plataforma y se utilizan en una gran variedad de escenarios de integración, dentro de las organizaciones y con socios de negocios.

Beneficios de SOA

Los beneficios de SOA para una organización se plasman a dos niveles distintos: al del usuario corporativo y a nivel de la organización de TI. Desde el punto de vista de la empresa, SOA permite el desarrollo de una nueva generación de aplicaciones dinámicas que resuelven una gran cantidad de problemas de alto nivel, fundamentales para el crecimiento y la competitividad.

SOA contribuye también a documentar e integrar el modelo de negocio de la empresa y dar y optimizar la respuesta a las dinámicas de cambio que se produzcan. La interfaz está completamente separada de la implementación. El software se proporciona estrictamente como un servicio que no tiene que ser

descargado e instalado. SOA promueve la reutilización y el compartir de servicios por numerosas aplicaciones e incluso por diferentes organizaciones.

Los productos Oracle SOA permiten crear, implementar y administrar SOA con la mejor tecnología integrada, ofreciendo:

- ***Una Plataforma SOA Completa y Previamente Integrada***—Conjunto completo de componentes para la infraestructura de procesos y servicios para crear, implementar y administrar SOA's.

- ***Administración de Ciclo Cerrado***—Administración completa del ciclo de vida de servicios

- ***Desempeño y escalabilidad excepcionales***—Transacciones en memoria, procesamiento de eventos en tiempo real y transferencia de datos de alto volumen sobre un servidor de aplicaciones altamente escalable

- ***Seguridad integrada***—seguridad centralizada, completa, de calidad empresarial, para la administración de políticas

8 TECNOLOGIAS

8.1 ORACLE

Oracle surge a finales de los 70 bajo el nombre de *Relational Software* a partir de un estudio sobre **SGBD** (Sistemas Gestores de Base de Datos) de George Koch. La tecnología Oracle se encuentra prácticamente en todas las industrias alrededor del mundo y en las oficinas de 98 de las 100 empresas *Fortune 100*. ***Oracle Corporation*** es una de las mayores compañías de software del mundo. Sus productos van desde bases de datos (Oracle) hasta sistemas de gestión. Cuenta además, con herramientas propias de desarrollo para realizar potentes aplicaciones, como Oracle Designer, Oracle JDeveloper y Oracle Developer Suite .

8.2 J2EE

Los departamentos de informática de las grandes empresas tuvieron la necesidad de crear un esquema que reformara sus redes y sistemas para dar servicio a miles de personas que de manera simultánea accedían a los recursos corporativos. Para atender estas expectativas, los técnicos tuvieron que redefinir la manera en que se almacenaba, se accesaba y se entregaba la información a los clientes.

Se hizo necesario la construcción de una arquitectura Cliente-Servidor multicapa, con el fin de compartir recursos entre clientes. Aparece la transmisión en tiempo real junto con el sistema operativo UNIX, el cual contiene soporte para el Protocolo de control de transmisión/ protocolo Internet (TCP/IP, *Transmission Control Protocol/ Internet Protocol*), Estándar que indica cómo crear, traducir y controlar transmisiones entre maquinas en una red de computadoras. Como consecuencia a lo anteriormente mencionado, surge la Llamada Remota a Procedimientos (*Remote Procedure Call o RPC*).

8.3 JDEVELOPER

Entorno de desarrollo integrado desarrollado por Oracle Corporation para los lenguajes Java, HTML, XML, SQL, PL/SQL, Java script, PHP, Oracle ADF, UML y otros. Es un software propietario pero gratuito desde 2005.

Las primeras versiones de 1998 estaban basadas en el entorno JBuilder de Borland, pero desde la versión 9i de 2001 está basado en Java, no estando ya relacionado con el código anterior de JBuilder.

Las últimas versiones estables son:

- Para JDK 6: 11.1.1.0.0 (abril de 2009)
- Para JDK 5: 10.1.3.5 (agosto de 2009).

Mejorar la productividad del personal de desarrollo, permitiendo responder de forma más rápida a los requerimientos de implementación de nuevos trámites. Oracle JDeveloper 10g es una herramienta de desarrollo Java integrada (IDE) que permite crear aplicaciones empresariales y Web Services de una manera rápida, sencilla y cumpliendo con los últimos estándares de la industria. Oracle JDeveloper 10g ofrece un soporte completo para el modelamiento, codificación, afinamiento, pruebas y deployment de aplicaciones empresariales.

JDeveloper permite que el desarrollo J2EE sea más simple y accesible ya que cuenta con un Framework (ADF) que incorpora un desarrollo visual y declarativo a la vez que implementa patrones de diseño para permitir a los programadores tener “productividad con alternativas”. Incluye además opciones avanzadas como: refactoring, control de versiones, modelamiento UML, desarrollo de base de datos.

8.4 ORACLE APPLICATION SERVER 10g

Oracle Application Server, un componente clave de Oracle Fusion Middleware, ofrece la solución más completa del sector para desarrollar, integrar e implementar las aplicaciones, portales y servicios Web de su empresa.

Basado en un poderoso servidor J2EE escalable, Oracle Application Server ofrece implementación de aplicaciones líderes, integración de negocios y el mejor software para portales. Oracle Application Server es la única plataforma diseñada para grid computing (inglés), así como soporte total del ciclo de vida de la Arquitectura Orientada a Servicios (SOA).

El servidor de aplicaciones Oracle ofrece funcionalidades empresariales extras que las herramientas *Open Source* no ofrecen como lo son:

1) Solución de portales

2) SOA / Automatización de Procesos

3) Seguridad

4) Administración

5) Herramienta de desarrollo integrada y Framework para asistencia al desarrollo

6) Solución de Inteligencia de Negocios

8.5 KEYTOOL

Si queremos enviar electrónicamente un documento importante, como un contrato, a otra persona, es una buena idea firmarlo "digitalmente", para que el receptor tenga una forma de comprobar que el documento realmente procede de nosotros y no fue alterado durante el tránsito.

Por defecto, la máquina virtual de Java dispone de las Entidades Certificadoras (CA) más comunes, como VeriSign o Thawte. Sin embargo, suele darse el caso, sobretodo en entornos de desarrollo, que necesitemos utilizar una Entidad Certificadora "de prueba". En este caso, debemos importar esta CA en el almacén de claves de la máquina virtual que estemos utilizando.
La máquina virtual de Java (JVM) cuenta con un almacén de claves (keystore) que incorpora las entidades más habituales y la posibilidad de agregar aquellas que nos sean necesarias. La base de datos que se utiliza en la Fiduciaria del Banco de Bogotá, es el TOAD de Oracle.

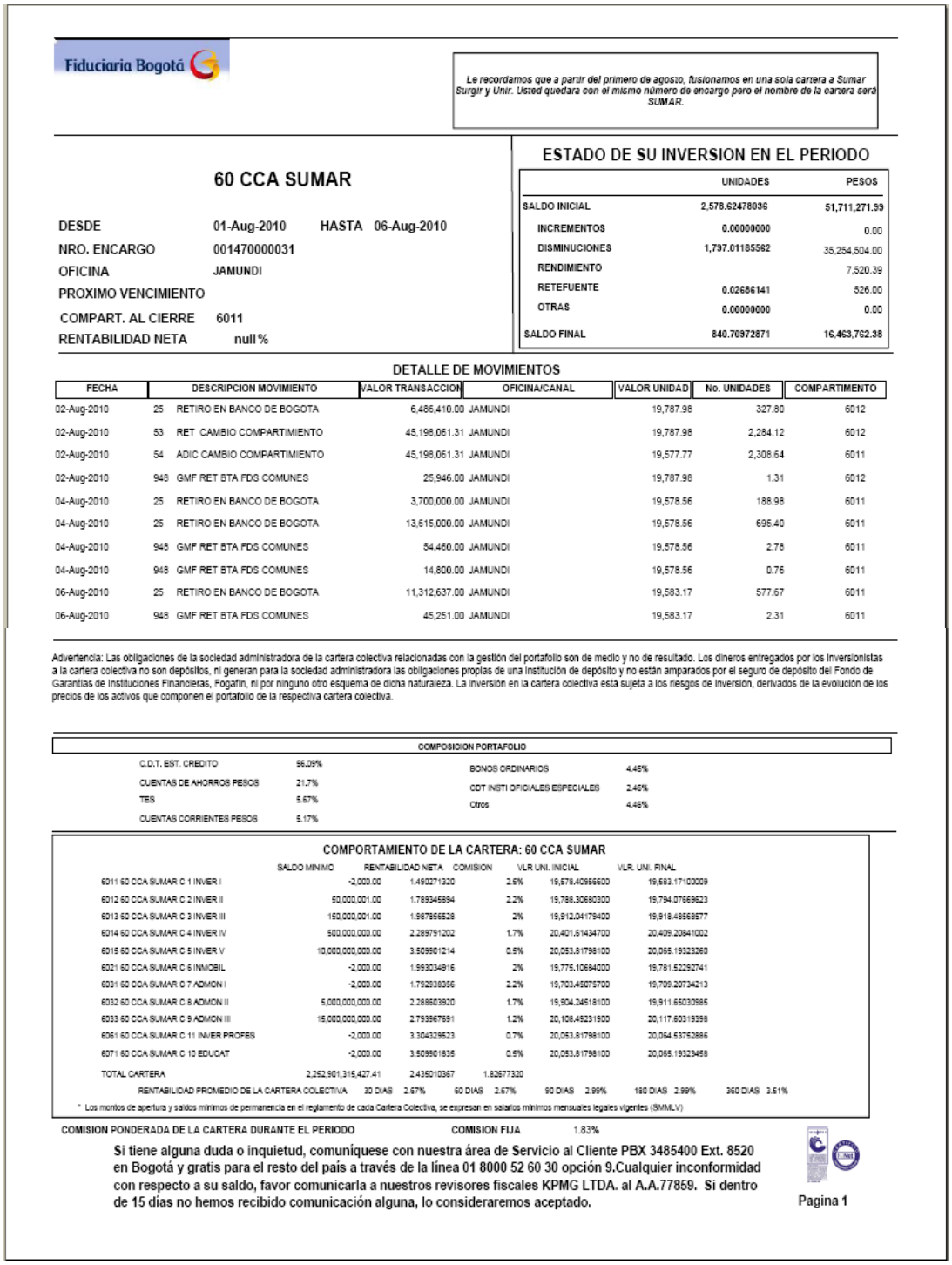
Para la simulación se Utilizará la base de datos de TOAD For MySQL.

Ilustración 20 Base de Datos Toad For MySQL

Para la generación del extracto se utilizo la herramienta ***iReport 1.2.7***

Ilustración 21 iReport 1.2.7

Con el que finalmente se genera el archivo .pdf con el extracto de un cliente en específico, que finalmente debe viajar seguro por la red, así:



Fiduciaria Bogotá

Le recordamos que a partir del primero de agosto, fusionamos en una sola carrera a Sumar Surgir y Unir. Usted quedara con el mismo número de encargo pero el nombre de la carrera será SUMAR.

60 CCA SUMAR

DESDE 01-Aug-2010 HASTA 06-Aug-2010
NRO. ENCARGO 001470000031
OFICINA JAMUNDI
PROXIMO VENCIMIENTO
COMPART. AL CIERRE 6011
RENTABILIDAD NETA null %

ESTADO DE SU INVERSION EN EL PERIODO

	UNIDADES	PESOS
SALDO INICIAL	2,578.62478036	51,711,271.99
INCREMENTOS	0.00000000	0.00
DISMINUCIONES	1,797.01185562	35,254,504.00
RENDIMIENTO		7,520.39
RETEFUENTE	0.02686141	526.00
OTRAS	0.00000000	0.00
SALDO FINAL	840.70972871	16,463,762.38

DETALLE DE MOVIMIENTOS

FECHA		DESCRIPCION MOVIMIENTO	VALOR TRANSACCION	OFICINA/CANAL	VALOR UNIDAD	No. UNIDADES	COMPARTIMENTO
02-Aug-2010	25	RETIRO EN BANCO DE BOGOTA	6,486,410.00	JAMUNDI	19,787.98	327.80	6012
02-Aug-2010	53	RET CAMBIO COMPARTIMIENTO	45,198,061.31	JAMUNDI	19,787.98	2,284.12	6012
02-Aug-2010	54	ADIC CAMBIO COMPARTIMIENTO	45,198,061.31	JAMUNDI	19,577.77	2,308.64	6011
02-Aug-2010	948	GMF RET BTA FDS COMUNES	25,946.00	JAMUNDI	19,787.98	1.31	6012
04-Aug-2010	25	RETIRO EN BANCO DE BOGOTA	3,700,000.00	JAMUNDI	19,578.56	188.98	6011
04-Aug-2010	25	RETIRO EN BANCO DE BOGOTA	13,615,000.00	JAMUNDI	19,578.56	695.40	6011
04-Aug-2010	948	GMF RET BTA FDS COMUNES	54,460.00	JAMUNDI	19,578.56	2.78	6011
04-Aug-2010	948	GMF RET BTA FDS COMUNES	14,800.00	JAMUNDI	19,578.56	0.76	6011
06-Aug-2010	25	RETIRO EN BANCO DE BOGOTA	11,312,637.00	JAMUNDI	19,583.17	577.67	6011
06-Aug-2010	948	GMF RET BTA FDS COMUNES	45,251.00	JAMUNDI	19,583.17	2.31	6011

Advertencia: Las obligaciones de la sociedad administradora de la cartera colectiva relacionadas con la gestión del portafolio son de medio y no de resultado. Los dineros entregados por los inversionistas a la cartera colectiva no son depósitos, ni generan para la sociedad administradora las obligaciones propias de una institución de depósito y no están amparados por el seguro de depósito del Fondo de Garantías de Instituciones Financieras, Fogafín, ni por ninguno otro esquema de dicha naturaleza. La inversión en la cartera colectiva está sujeta a los riesgos de inversión, derivados de la evolución de los precios de los activos que componen el portafolio de la respectiva cartera colectiva.

COMPOSICION PORTAFOLIO

C.D.T. EST. CREDITO	56.09%	BONOS ORDINARIOS	4.45%
CUENTAS DE AHORROS PESOS	21.7%	CDT INSTI OFICIALES ESPECIALES	2.46%
TES	5.67%	Otros	4.46%
CUENTAS CORRIENTES PESOS	5.17%		

COMPORTAMIENTO DE LA CARTERA: 60 CCA SUMAR

	SALDO MINIMO	RENTABILIDAD NETA	COMISION	VLR UNI. INICIAL	VLR. UNI. FINAL
6011 60 CCA SUMAR C 1 INVER I	-2,000.00	1.490271320	2.5%	19,578.40956600	19,583.17100009
6012 60 CCA SUMAR C 2 INVER II	50,000,001.00	1.789346894	2.2%	19,788.30680300	19,794.07669623
6013 60 CCA SUMAR C 3 INVER III	150,000,001.00	1.987866528	2%	19,912.04179400	19,918.48568577
6014 60 CCA SUMAR C 4 INVER IV	500,000,000.00	2.289791202	1.7%	20,401.61434700	20,409.20841002
6015 60 CCA SUMAR C 5 INVER V	10,000,000,000.00	3.509901214	0.5%	20,053.81798100	20,065.19323260
6021 60 CCA SUMAR C 6 INMOBIL	-2,000.00	1.993034916	2%	19,775.10684000	19,781.52292741
6031 60 CCA SUMAR C 7 ADMON I	-2,000.00	1.792938366	2.2%	19,703.45075700	19,709.20734213
6032 60 CCA SUMAR C 8 ADMON II	5,000,000,000.00	2.288603920	1.7%	19,904.24618100	19,911.65030965
6033 60 CCA SUMAR C 9 ADMON III	15,000,000,000.00	2.793967691	1.2%	20,108.49231900	20,117.60319398
6061 60 CCA SUMAR C 11 INVER PROFES	-2,000.00	3.304329523	0.7%	20,053.81798100	20,064.53762886
6071 60 CCA SUMAR C 10 EDUCAT	-2,000.00	3.509901835	0.5%	20,053.81798100	20,065.19323458
TOTAL CARTERA	2,252,901,315,427.41	2.435010367	1.82677320		

RENTABILIDAD PROMEDIO DE LA CARTERA COLECTIVA 30 DIAS 2.67% 60 DIAS 2.67% 90 DIAS 2.99% 180 DIAS 2.99% 360 DIAS 3.51%

* Los montos de apertura y saldos mínimos de permanencia en el reglamento de cada Cartera Colectiva, se expresan en salarios mínimos mensuales legales vigentes (SMMLV)

COMISION PONDERADA DE LA CARTERA DURANTE EL PERIODO COMISION FIJA 1.83%

Si tiene alguna duda o inquietud, comuníquese con nuestra área de Servicio al Cliente PBX 3485400 Ext. 8520 en Bogotá y gratis para el resto del país a través de la línea 01 8000 52 60 30 opción 9.Cualquier inconformidad con respecto a su saldo, favor comunicarla a nuestros revisores fiscales KPMG LTDA. al A.A.77859. Si dentro de 15 días no hemos recibido comunicación alguna, lo consideraremos aceptado.

Pagina 1

Ilustración 22 EXTRACTO FINAL CLIENTE

9 APLICACIÓN WSEXTRACTO

El proyecto de investigación propuso la implementación para la aplicación denominada **WSExtracto**, por medio de la cual se generarían extractos bancarios transmitidos por el canal público, que comunica a la Fiduciaria del Banco de Bogotá con el bus transaccional de ATH. Como primera medida la temática a indagar se hizo en torno a la creación de Web Services: ¿Cuáles son los componentes? ¿Cómo se describen, publican, localizan y prestan sus servicios? ¿Qué Riesgos de seguridad presentan?, entre otras.

Seguidamente se hizo relación al tema de la Arquitectura Orientada a Servicios, con lo que se busco profundizar, contextualizar y coordinar las partes que componen el Servicio Web. El modulo de seguridad, fundamentalmente fue el reto para que este proyecto hiciera parte de las aplicaciones que a diario se manejan por parte de Flexiweb, especialmente para el aplicativo de **FondosWEB**, siendo este el pionero de muchos otros que actualmente se proponen en la Fiduciaria de Bogotá. Personalmente, conformar el equipo de desarrollo de Flexiweb LTDA, no solo constituía la oportunidad de recibir el galardón como ingeniera sino también el crecer laboral y personalmente.

El requisito primordial para que la comunicación entre las dos entidades Bancarias fuese un éxito, se resume en la utilización de Certificados X.509, los cuales garantizarán la seguridad en el envió de los extractos. Recordando que los Web Services permiten la utilización de herramientas de Software variadas sin que estas influyan en los desarrollos de parte y parte de los entes involucrados en este proyecto.

9.1 *DEFINIENDO HERRAMIENTAS*

Los aplicativos están desarrollados en lenguaje de Programación JAVA y PL/SQL. Como primer obstáculo por solucionar, requería la búsqueda de una version de JDeveloper más adelante de la que se tenía, que soportará el modulo de seguridad y que a su vez se pudiese implementar en el Servidor de aplicaciones OAS.

Lo anterior solicitaba renovación tecnológica, tanto en el IDE de desarrollo como en el Servidor. ATH propuso la utilización de la herramienta OpenSSL. (Popular paquete open source, de administración y librerías sobre criptografía), que permite a un atacante obtener el componente privado de una clave RSA. Científicos de la Universidad de Michigan, encontraron que podían deducir pequeñas piezas de una clave privada mediante la inyección de pequeñas fluctuaciones en el suministro de energía de un dispositivo mientras realiza el procesamiento de mensajes cifrados. En poco más de 100 horas, manipulando la alimentación del dispositivo, generaron suficientes "fallos transitorios" para reunir la totalidad de su clave de 1024 bits

Esencialmente por librerías el OAS no soporto este tipo de Llaveros así que se dejo a un lado esta herramienta. Lo siguiente, consistió en la adquisición del JDeveloper 10.1.3.4 quien afortunadamente solucionaba gran parte de los problemas, al incluir la herramienta para la creación de llaves "Keytool" además del certificado X.509 solicitado por el Cliente.

La idea global del proyecto, identifica 4 actores importantes para la creación de WSExtracto, que son:

- ***Web Browser aplicación cliente***

  Es la interfaz gráfica que permite la interacción del usuario con el sistema, en la cual el usuario puede seleccionar el nivel de seguridad con el que desea trabajar y realizar las acciones de administración de los derechos electrónicos.

- ***Web Services Fachada del Sistema***

  Es el punto de entrada a la lógica del sistema, que administra la comunicación entre el cliente y los diferentes WS. Cuando así se requiere solicita los tokens de seguridad al WSSecurity (para realizar la autenticación y federación), y administra las llamadas al Web Services "**WSExtracto**" de la fiduciaria del Banco de Bogotá y el de ATH.

- ***Web Services de Seguridad***

  Es el WS que se encarga de la administración y emisión de Tokens de Seguridad con el fin de garantizar la autenticación de usuarios en el sistema

y crear un contexto federado que establece un conjunto de participantes con relaciones de confianza implícitas entre ellos.

- ***Web Services WSExtracto (Fiduciaria Banco de Bogotá)***

  Es el responsable de administrar los derechos electrónicos que se encuentran almacenados en la base de datos de la Fiduciaria. Se encarga de llevar a cabo las operaciones de Consulta y generación de Extractos Bancarios.

La creación de las llaves es el paso importante para el despliegue correcto del WS en el servidor y la respuesta optima al llamado del cliente. Como se indico, JDeveloper incluye la herramienta *Keytool.* Ubicada en la ruta **<JDEV_HOME>**jdevstudio10134\jdk\bin, en la cual se encuentra instalado el JDeveloper de nuestros equipos. Para el efecto de la creación de llaveros, se ejecutan comandos que la herramienta reconoce y que incluyen los algoritmos de cifrado RSA-SHA1, RSA-MD5.

9.2 *CREACIÓN DE LLAVES*

La creación del llavero, implica la creación de llaves tanto para el servidor y para el Cliente.

1. Servidor.jks es nuestro almacen de llaves principal para WSExtracto, el cual consta de la llave de firma “ServerSIGN” y la llave de encriptación “ServerENC” , ambas para el servidor.

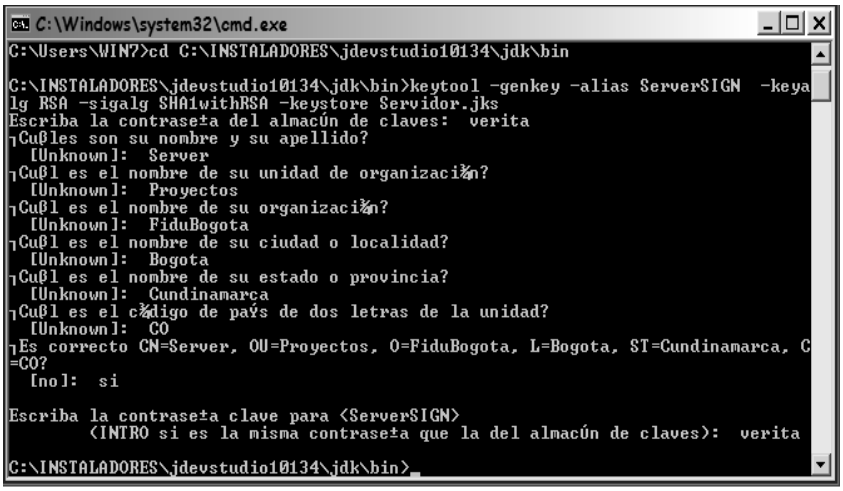


Ilustración 23 ServerSIGN

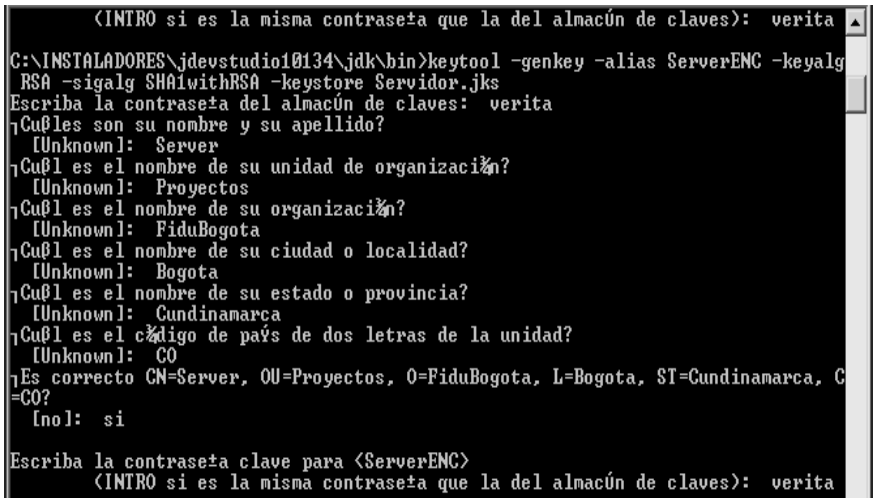


Ilustración 24 ServerENC

2. Se crea el almacen de llaves para el Cliente, ClienteATH.jks

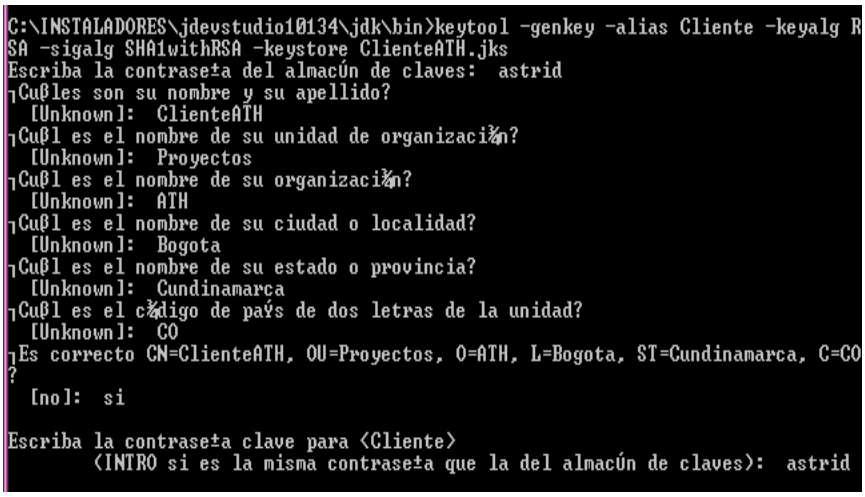


Ilustración 25 ClienteATH.jks

Finalmente, se incluyen los certificados para cada una de las llaves del servidor como se observa a continuación y en los que se evidencia la utilización de los algoritmos soportados por JDeveloper MD5 y SHA1.

Se deben copiar las llaves públicas del servidor al cliente y del cliente al servidor. Para exportar la llave de encripción del servidor, se tiene:

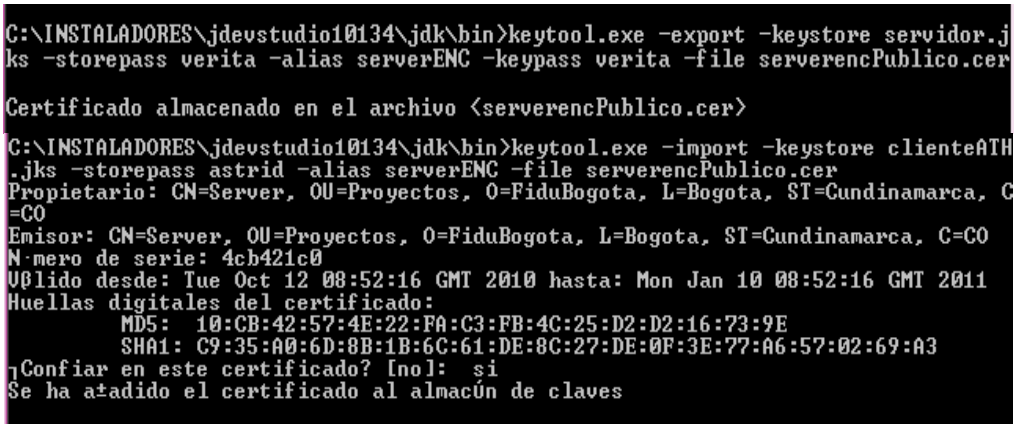


Ilustración 26 Importando Certificados.

Se hace la importación de la llave de encripción del servidor en el almacén de claves del cliente. Primero se exportando el certificado del cliente y

```
C:\INSTALADORES\jdevstudio10134\jdk\bin>keytool.exe -export -keystore ClienteATH
.jks -storepass astrid -alias cliente -keypass astrid -file clientePublico.cer
Certificado almacenado en el archivo <clientePublico.cer>
```

Ilustración 27 Exportando Certificados al cliente

Y finalmente, se importa en el almacén del servidor.

```
C:\INSTALADORES\jdevstudio10134\jdk\bin>keytool.exe -import -keystore servidor.j
ks -storepass verita -alias cliente -file clientePublico.cer
Propietario: CN=ClienteATH, OU=Proyectos, O=ATH, L=Bogota, ST=Cundinamarca, C=CO

Emisor: CN=ClienteATH, OU=Proyectos, O=ATH, L=Bogota, ST=Cundinamarca, C=CO
N·mero de serie: 4cb4227d
Vßlido desde: Tue Oct 12 08:55:25 GMT 2010 hasta: Mon Jan 10 08:55:25 GMT 2011
Huellas digitales del certificado:
         MD5:  FD:74:8E:A3:A6:D1:43:4C:36:48:CB:D4:34:A2:5A:82
         SHA1: 4C:1A:AA:B3:5A:4A:FD:BC:D7:31:58:53:7E:7D:9C:46:64:B4:58:AF
¬Confiar en este certificado? [no]:  si
Se ha a±adido el certificado al almacÚn de claves

C:\INSTALADORES\jdevstudio10134\jdk\bin>keytool.exe -list -v -keystore Servidor.
jks
Escriba la contrase±a del almacÚn de claves:  verita

Tipo de almacÚn de claves: jks
Proveedor de almacÚn de claves: SUN
```

Ilustración 28 Importando Certificados al Servidor

En la ruta de ejecución en que ejecutamos los comandos, encontramos los almacenes y certificados X.509 creados para cada uno de las llaves del Servidor.

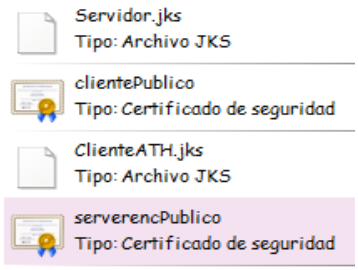


Ilustración 29 Almacenes de Llaves y Certificados creados

Keytool.exe -list -v -keystore **<JDEV_HOME>**\jdevstudio10134\jdk\bin \servidor.jks

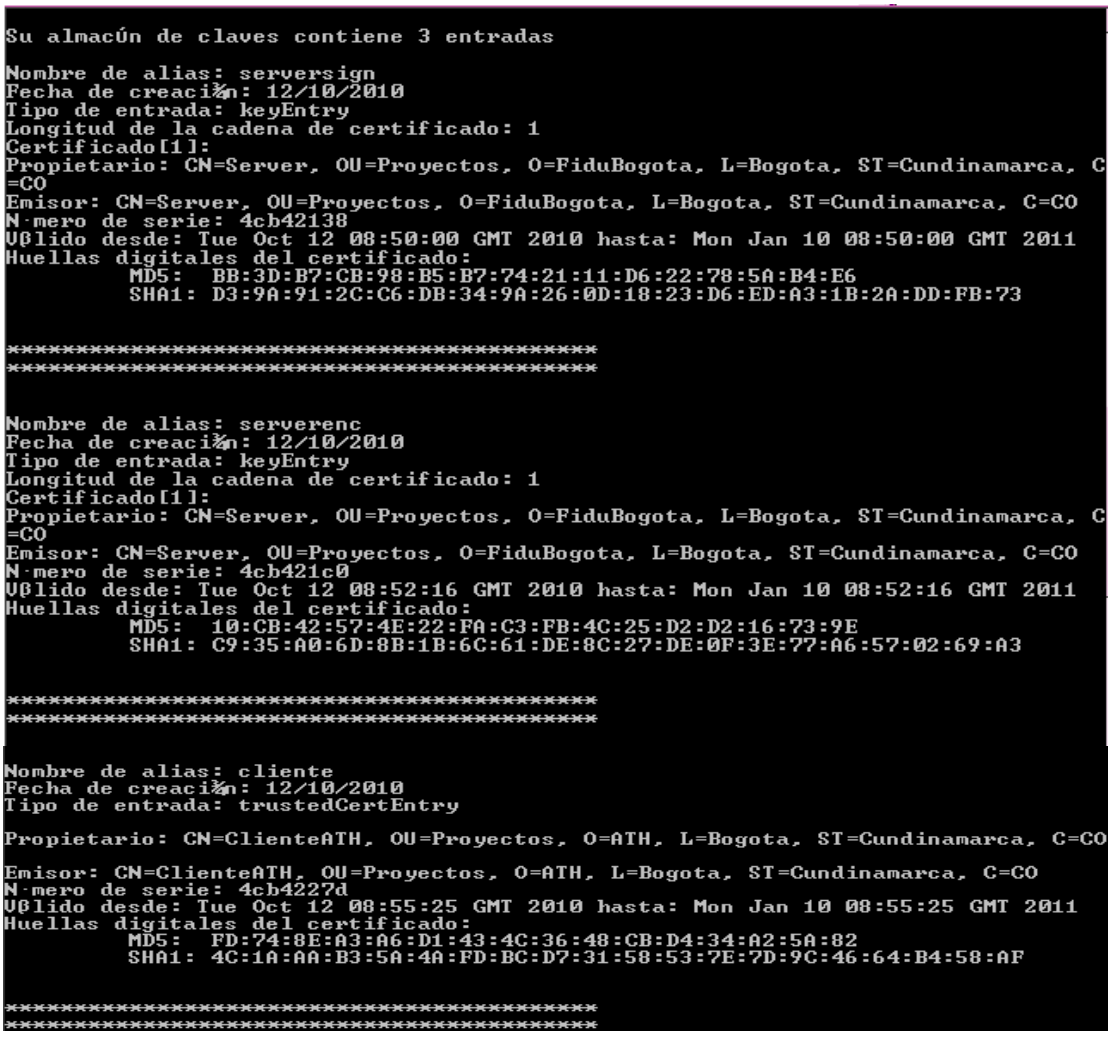


Ilustración 30 Listando Servidor.jks

keytool.exe -list -v -keystore **<JDEV_HOME>**jdevstudio10134\jdk\bin \cliente.jks

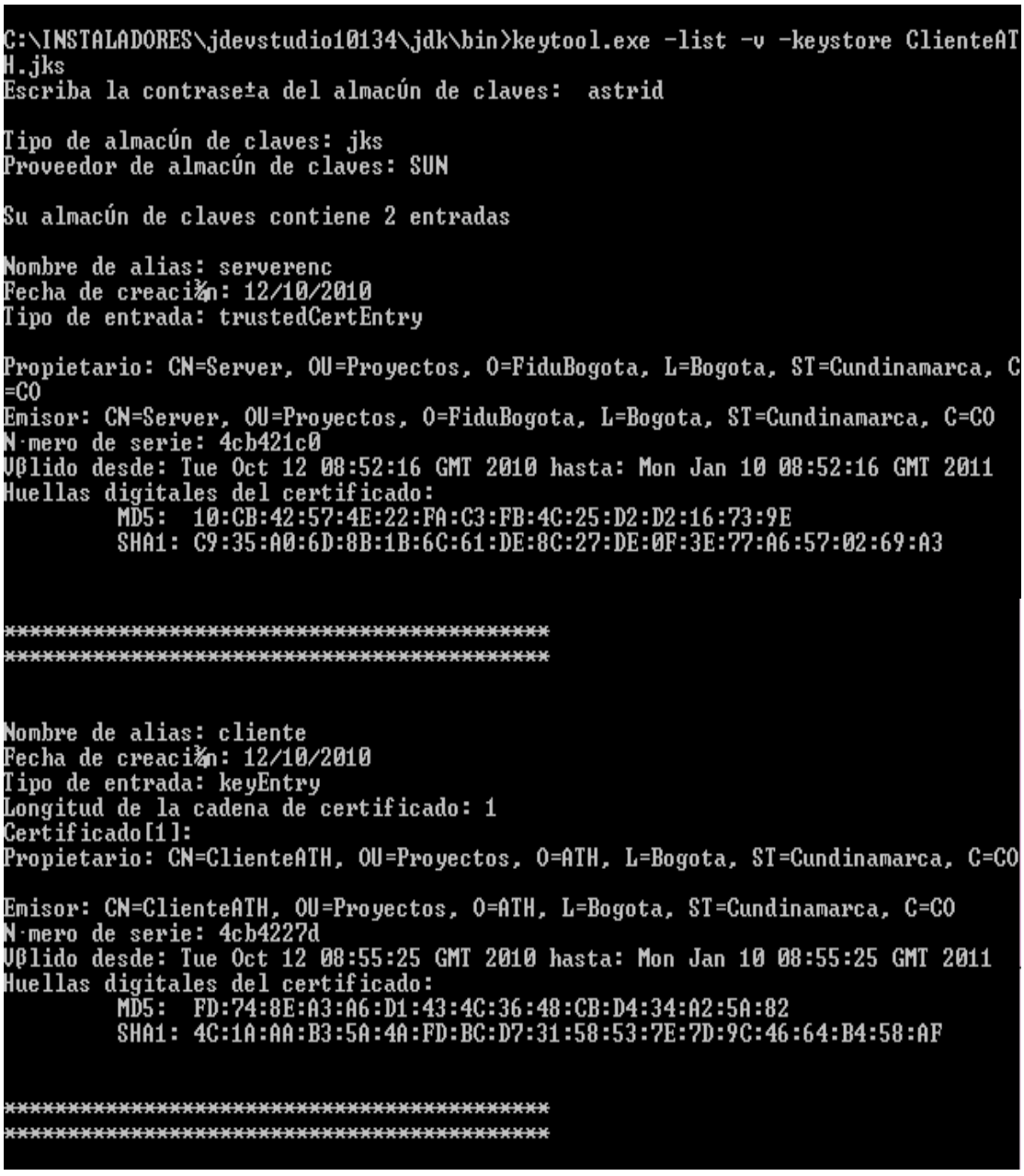


Ilustración 31 Listando ClienteATH.jks

Para la creación del Web Services, JDeveloper cuenta con asistentes que facilitan estas tareas.

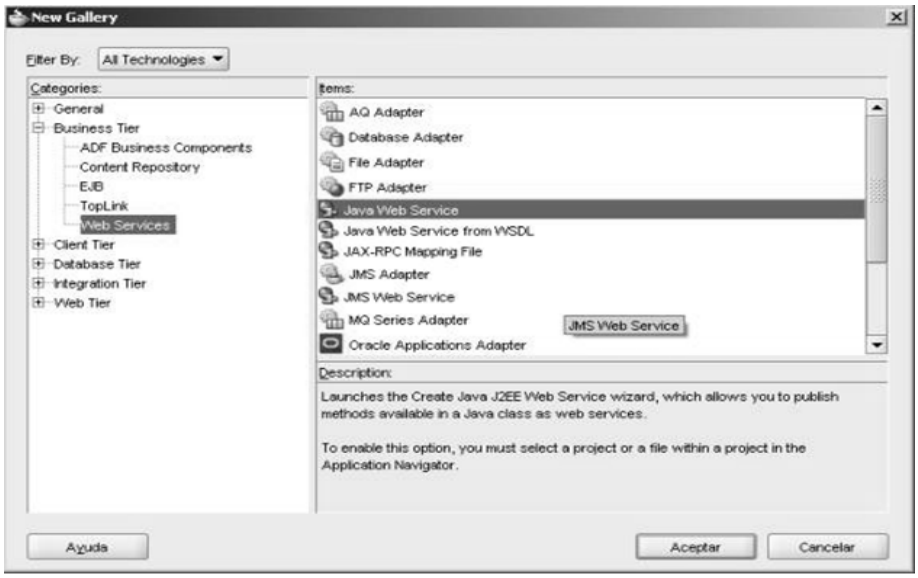


Ilustración 32 Wizard JDeveloper 10.1.3.4

Para la implementación de la seguridad, el Wizard presenta un modulo de seguridad, así:

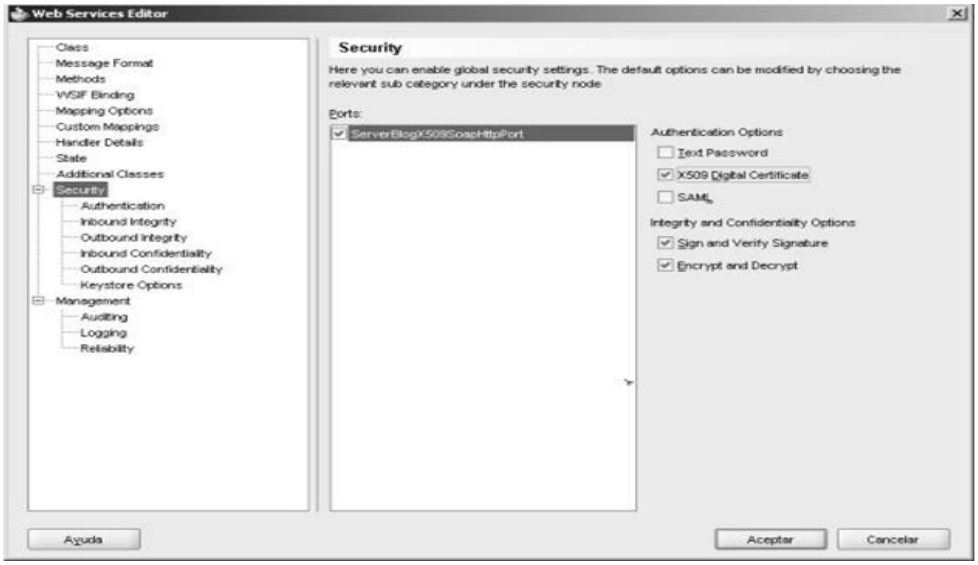


Ilustración 33 Editor Modulo de Seguridad JDeveloper 10.1.3.4

Es importante que desde Java como desde el modulo de Administración del Servidor se configure la parte de seguridad, puesto que una vez desplegado el servicio este responde a cualquier llamado si y solo si las llaves de firmado y encriptación están correctamente implementadas.

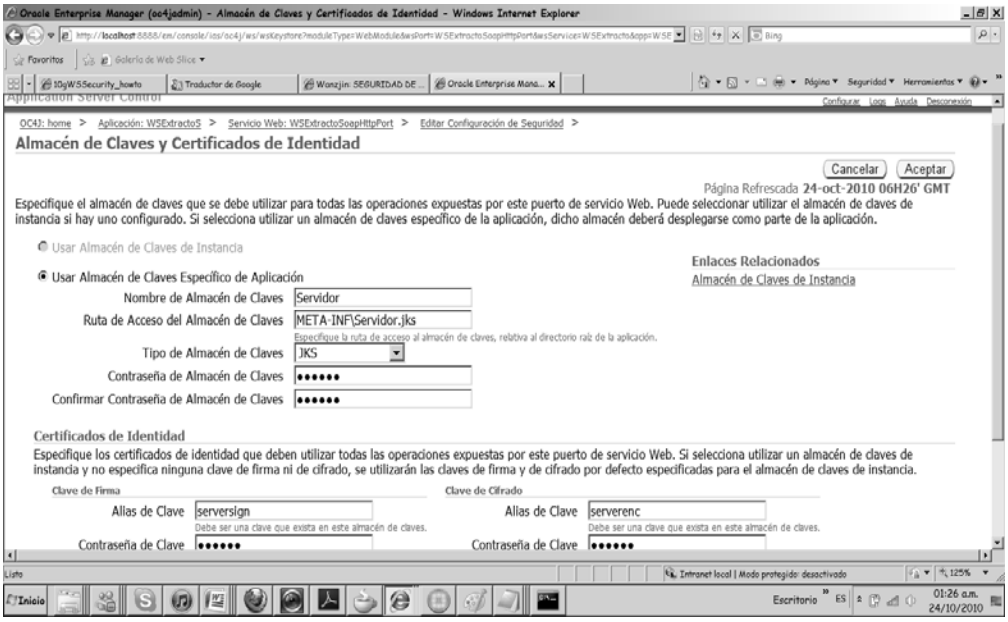


Ilustración 34 Editor Modulo de Seguridad OC4J

Localmente, el HTTPAnalyzer, herramienta del JDeveloper, permite visualizar la respuesta del servidor al llamado de un cliente.

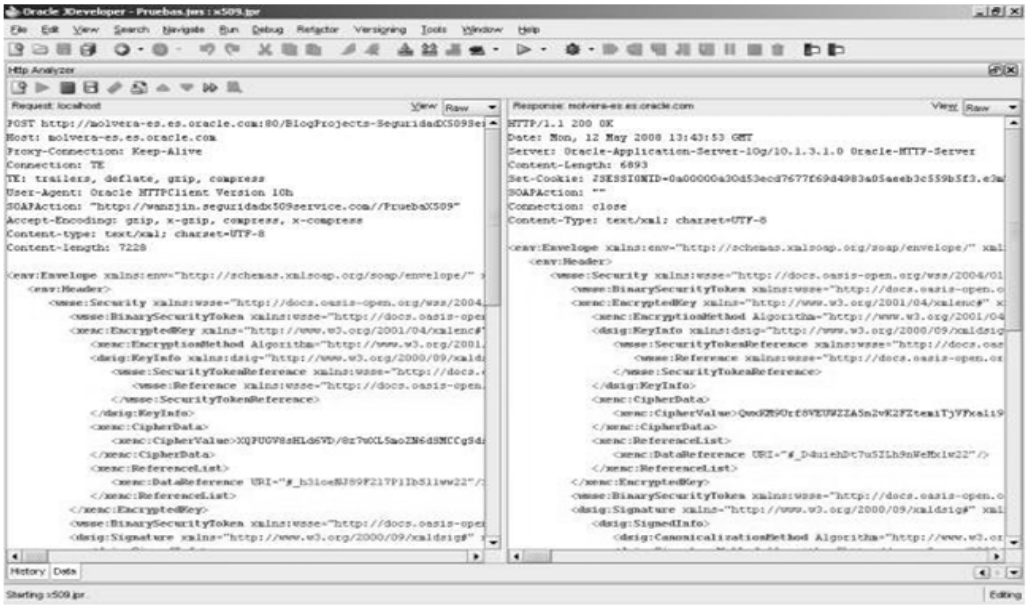


Ilustración 35 HTTPAnalyzer JDeveloper 10.1.3.4

10 INCONVENIENTES PRESENTADOS EN LA IMPLEMENTACION DE LA SEGURIDAD

El último paso que dio este Proyecto de investigación fue realizar un conjunto de pruebas para dar por completados y satisfechos los objetivos planteados. Las pruebas debieron limitarse a la medición del rendimiento, confiabilidad y disponibilidad del sistema en términos de tiempo de respuesta a las solicitudes del usuario y por otra parte la evaluación de la confidencialidad e integridad de la información que viaja entre los Web services mediante el uso de un analizador de protocolos de red.

1. Es importante configurar las contraseñas correctamente del lado del cliente. De ser erróneo presentará un error, así: "***Keystore was tampered with, or password was incorrect."***

2. Es posible que la configuración del servidor en la administración del servicio sea incorrecta. Se debe configurar el usuario con su respectiva contraseña, quien será quien ejecute la operación para la cual fue creado el Servicio. Este error se evidencia "***An invalid token was provided".***

3. Existe un término importante en el despliegue de los servicios conocido como "***TIMESTAMP***", este atributo maneja la sincronía entre las máquinas utilizadas para el servicio. Este atributo debe coincidir tanto en el servidor, como en el cliente. Su valor nunca debe estar "False".

   Como parte de los inconvenientes técnicos entre las máquinas Cliente-Servidor se dedujo a partir de la investigación que el reloj de la máquina del cliente debe estar atrasado con respecto al del servidor.

4. Se presentaron errores de Conexión tanto para acceder al servicio, como a la conectarse a la base de datos. Esto se daba en la medida en que la base de datos estaba abajo y no se tenía acceso a los ambientes de pruebas o desarrollo. Esto sucede similarmente con el servidor de aplicaciones donde se encuentra desplegado el servicio.

   En ocasiones se encuentran sesiones Zombies que evitan la ejecución de tareas. Por tanto se debe reiniciar el contenedor del OAS o la base de datos en conflicto (desarrollo, pruebas o producción).

5. Las validaciones de los errores, se hicieron en la medida que los cajeros al generar los extractos, supiesen exactamente en que estaban fallando. Por ejemplo al incluir un formato de fechas incorrecto. En nuestro caso debería ser dd-mm-yyyy, de no ser así arroja el siguiente error

"005 - El Formato de una de las fechas no es válido o esta vacio"

Esto mismo se hizo con cada uno de los campos que el servidor necesita para generar el extracto

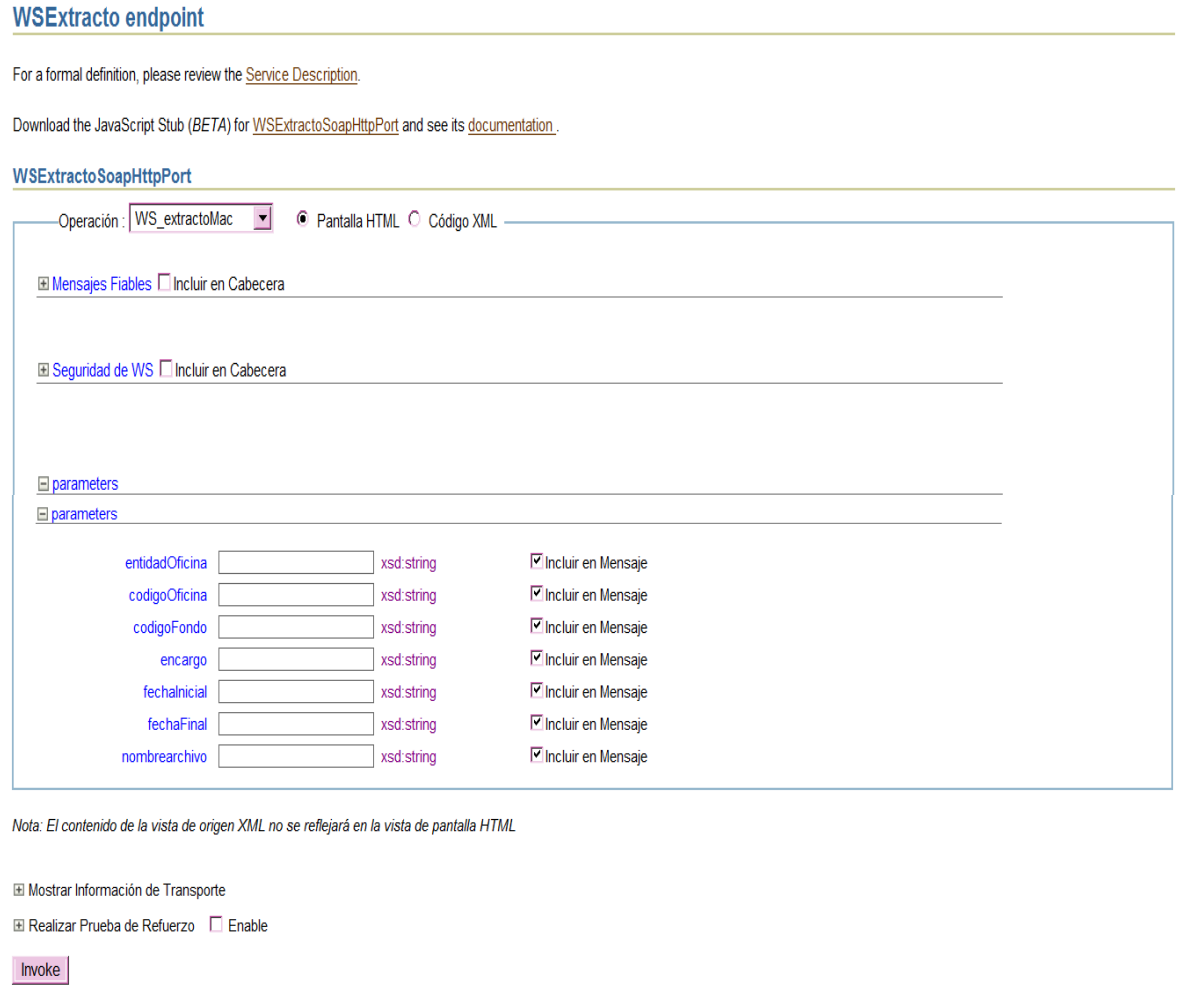


Ilustración 36 Interfaz WSExtracto

Algunos extractos fueron corregidos, ya que la información extraída de la base de datos no correspondía con la que contaba realmente los clientes de pruebas. Esto hace parte más de la lógica del negocio.

11 ACTIVIDADES ADICIONALES FIDUCIARIA BOGOTA

- ✓ Responsable HelpDesk en la Fiduciaria del Banco de Bogotá (FiduBogotá).
- ✓ Coordinadora de nuevos requerimientos que la Fiduciaria haga a la empresa.
- ✓ Backup de la Base de datos de desarrollo.
- ✓ Responsable Paso a Pruebas de los nuevos desarrollos realizados.
- ✓ Generación de aplicación WAR de los aplicativos FondosWeb e InverWeb a pruebas y producción.
- ✓ Documentación Paso a Pruebas y Producción aplicativos FondosWeb e Inverweb.
- ✓ Gestión de los proyectos con proveedores del área de sistemas.
- ✓ Soporte y desarrollo de aplicaciones de administración de fondos y de portafolios de inversión.

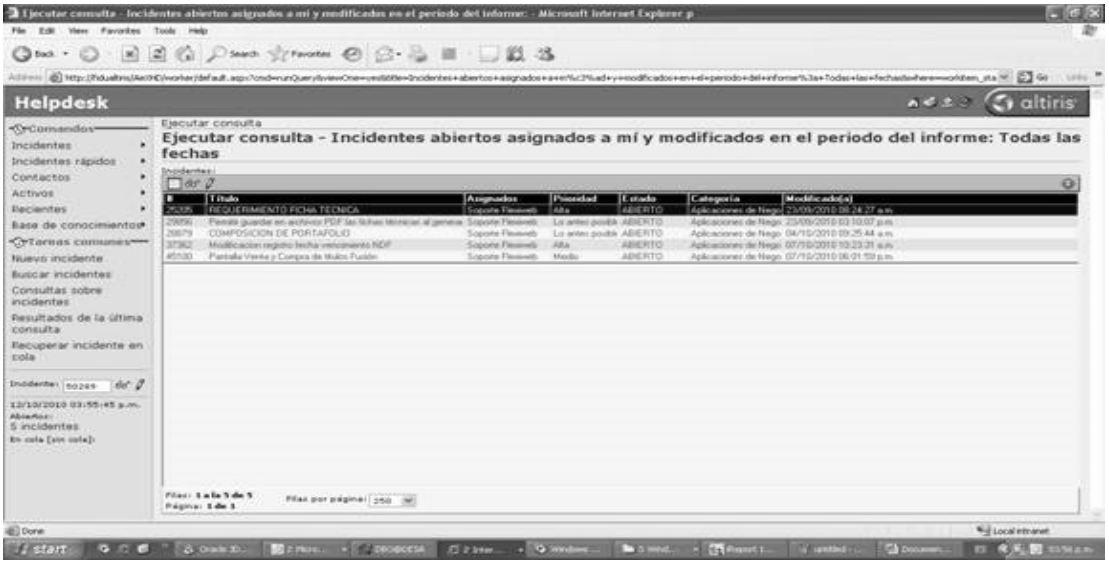


Ilustración 37 ALTIRIS

Fiduciaria Banco de Bogotá, maneja Software para HelpDesk conocido como **ALTIRIS,** por medio del cual se tratan los incidentes o mejoras que presentan los usuarios, para lo que crean la respectiva **MESA DE AYUDA** (MA), describiendo el

objetivo de la misma, sea corrección de incidentes en el ambiente de producción o la creación de nuevas mejoras en los aplicativos. Creadas generan un nuevo identificador ***Id*** con el cual se hace seguimiento para cada caso.

Existe un filtro conocido como ***Soporte Aplicativos,*** quien conoce los aplicativos que posee la empresa. Encargado de validar los errores de Data para el ambiente de producción. Cuando no se tiene certeza de lo que ocurre, procede con la asignación de la MA al respectivo proveedor para ser atendida en el menor tiempo posible.

Las mejoras, requieren de cotización, partiendo del Análisis y Diseño de cada mejora de acuerdo al formato de requerimiento entregado por el usuario, "*lo escrito, escrito esta*". Una vez aprobadas por el comité funcional de la vicepresidencia de tecnología se procede con el desarrollo, posteriormente se realizan las pruebas de escritorio para generar el ***Paso a Pruebas,*** para***.*** Este último puede ser de tres tipos: El primero, solo para objetos a compilar en la base de datos ya sea para el esquema DBO (fondos), Inverweb(inversiones) o Fuerza de Ventas.(Mercadeo). El segundo el despliegue en el OAS para el ambiente de Pruebas de un archivo comprimido conocido como WAR, dependiendo del aplicativo al que corresponde la mejora. Y finalmente, Se puede hacer paso a pruebas tanto para la Base de datos como para el despliegue de WAR. Para realizar el seguimiento de las pruebas de usuario y de las mejoras, existe el filtro conocido como **Soporte a Pruebas.**

Una vez que los usuarios que realizan las pruebas indican que el ***DECK de pruebas,*** es exitoso, se procede al paso a producción de los objetos de Base de datos o del despliegue del War para actualizar el aplicativo en la Fiduciaria. Esta actividad la coordina *soporte aplicativos.*

Para cada uno de los ambientes, ya sea el de Desarrollo, Pruebas o Producción es importante mantener la Bitácora de los objetos de Base de datos, entre los que se tienen los Procedimientos, funciones, tablas, entre otros. Puesto que en ocasiones se hacen actualizaciones con pérdida de objetos. Esta actividad se hace constantemente para cada mes del año.

12 CONCLUSIONES

.

- El modelo de Seguridad propuesto analiza las vulnerabilidades de estas aplicaciones Web e identifica, establece, describe e implementa un anillo de seguridad de tipo preventivo, detectivo y correctivo para soportar los siete principios de seguridad que son el ideal de la máxima seguridad.

- El futuro de las aplicaciones Web es bastante promisorio, ya que cada vez más se difunde entre las empresas, universidades, pequeños y medianos negocios, la necesidad de tener presencia en la Web, conforme el número de cibernautas se incrementa, también lo hace el número de clientes y usuarios de dichas aplicaciones.

- Hoy en día, las aplicaciones Web basan sus esquemas de seguridad, principalmente en el uso de la criptografía tanto simétrica como asimétrica, por lo cual mientras los algoritmos criptográficos utilizados (SHA, RSA, AES, etc.) no sean vulnerados, podemos confiar que toda la información de nuestro sistema Web goza del beneficio de ser confidencial, íntegra y autenticada; No obstante, a medida que la tecnología evoluciona, el criptoanálisis a los algoritmos puede llegar a hacerlos vulnerables en algún momento.

- Los WS, aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen. Fomentando los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.

- Al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado. Permitiendo que los servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.

- Permiten la interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándar y abiertos. Las especificaciones son gestionadas por una organización abierta, la W3C, por tanto no hay secretismos por intereses particulares de fabricantes concretos y se garantiza la plena interoperabilidad entre aplicaciones.

- Con el uso de JDeveloper es posible crear, publicar, asegurar y desplegar servicios Web a los estándares WS-Security. JDeveloper se puede utilizar como banco de pruebas para llamar a los servicios desplegados con toda la seguridad necesaria y habilitado para ver los resultados en el analizador de HTTP. Para desplegar los servicios se debe realizar por medio de un Servidor de aplicaciones con privilegios de seguridad para administrar de forma centralizada los perfiles de seguridad y gestión de los servicios.

- Teniendo en cuenta criterios de evaluación algunas recomendaciones son:

  - Si la infraestructura de la organización es pequeña, el presupuesto corto y la aplicación no requiere un alto porcentaje de disponibilidad, se puede implementar un balanceador de carga de software.

  - Si la infraestructura de la organización es considerable, el presupuesto amplio y la aplicación requiere un alto porcentaje de disponibilidad, es aconsejable utilizar un balanceador de carga de hardware.

- Cuando el usuario selecciona la opción “iniciar Sesión” y digita su *Username* y su *Password,* el sistema valida los datos ingresados por el usuario y si están bien, retorna información del usuario entre la que se encuentra el rol que este tiene en el sistema, el cual puede ser de dos tipos, administrador o usuario; de esta forma, dependiendo del rol que tenga, se prepara el portafolio de usuario o administrador con sus respectivas opciones. El rol de la persona que ingresa al sistema se guarda como una variable de sesión para manejar la interfaz gráfica que debe mostrarse al usuario.

- Cabe aclarar que no se puede tener acceso directo a los portafolios tanto de USUARIO como ADMINISTRADOR sin haber iniciado sesión previamente y que sólo desde estas páginas se pueden tener acceso a las opciones de consulta de Extractos.

- En este proyecto de investigación se implemento WSExtracto, utilizando un token de seguridad (certificado X.509) emitido por la Entidad de Seguridad del mismo IDE de desarrollo JDeveloper 10g. Encargada de crear y emitir el mismo token de seguridad a todos aquellos participantes que ella incluye

en una lista de suscritos. Finalmente el cliente obtiene el acceso al extracto que solicitado y el paso siguiente será la validación de la información que WSExtracto le respondía.

- Cada vez que un participante solicita por primera vez al WS de fachada el acceso a un servicio, la fachada solicita al Web services de seguridad una suscripción a lo que se responde con el token de WSExtracto. Las próximas solicitudes, el cliente no deberá autenticarse con una firma ya que se entabla de manera implícita una relación de confianza.

- Si el cliente solicita la renovación del token de seguridad del WSExtracto, la Fachada emite un mensaje a la entidad responsable y esta se encargará de eliminar la suscripción y crear una nueva, a lo que responde con un token con similares características excepto por su vigencia.

- La confidencialidad, también conocida como privacidad, es el proceso de asegurarse, que los datos sensibles o privilegiados permanezcan privados y confidenciales y por consiguiente no pueden ser entendidos por usuarios sin autorización o por personas que monitorean el flujo de tráfico a través de la red (uso de Sniffers).

- La encripción es la técnica usada para garantizar la confidencialidad en esta aplicación EL frameWork JDeveloper permite a clientes y WS creados usar Keystore de Java. Encriptar y desencriptar mensajes SOAP, los cuales por defecto viajan por la red en texto plano y por lo tanto pueden ser vistos y entendidos por cualquier persona, las técnicas que se utiliza para tal fin es la encripción asimétrica, donde un cliente puede encriptar el mensaje utilizando la llave pública del certificado X.509 del usuario final (receptor del mensaje) para garantizar que solo el dueño de la llave privada de dicho certificado X.509 pueda desencriptar el mensaje.

  Por otra parte, la encripción simétrica requiere que tanto el emisor como el receptor del mensaje compartan una llave secreta para con ella encriptar y desencriptar el mensaje SOAP.

- La integridad es el principio que garantiza que los datos estén protegidos contra modificaciones accidentales o deliberadas. La integridad, se garantiza mediante la implementación de firmas digitales con XML; mediante el uso del XML signature y certificados X.509. Los certificados por

ser de prueba limitaron las pruebas de este principio ya que ninguna autoridad certificadora se presta para validar y autenticar su legitimidad.

- El principio de NO REPUDIO se debe implementar en un sistema de WS teniendo en cuenta las siguientes consideraciones:

  - Se debe verificar la identidad del usuario a través de un certificado digital u otro medio que la garantice.
  - Se deben registrar las operaciones críticas realizadas en el sistema de tal forma que reflejen información clave para el no repudio como identificación del usuario, fecha y hora de la operación, entre otros.

  Por estas dos razones se hace uso de las firmas digitales como medio para garantizar la autenticidad del usuario y el registro del Log de auditoría para establecer exactamente el ámbito en el que se ha realizado cierta operación

- La seguridad informática es supremamente extensa y profunda, y también poco aplicada alrededor del mundo, no solo en nuestro país. Es por esto, que así como este proyecto de investigación culmina, quedan abiertas las puertas e invito a nuevos estudiantes a vincularse con un tema de fuerte potencial en el mercado global.

- La practica empresarial, es la oportunidad perfecta para que muchos estudiantes, se induzcan en la vida laboral. Es propicio fomentar la disciplina y la investigación con el fin desempeñarse exitosamente al afrontar retos importantes en las empresas que brindan esta ayuda. Que el no conocer sea impedimento para dejar a un lado los proyectos que se impongan

13 RECOMENDACIONES

1. Se recomienda para futuros desarrollos tener encuenta las tecnologías con las que se cuentan, de tal forma que las aplicaciones dejen de ser consideradas como Modelos de prototipo, y pasen a ser aplicaciones que corren en ambientes de producción óptimos

2. El desarrollo de una práctica empresarial es una experiencia satisfactoria además de enriquecedora que promueve el desarrollo y el aprendizaje. La incursión en un ambiente laboral permite reconocer y valorar muchas fortalezas adquiridas a lo largo del proceso académico, así como las falencias que se tienen y que requieren de un gran esfuerzo para mejorar. Es entonces la oportunidad para que muchos otros estudiantes puedan vivir este tipo de experiencias.

3. Se recomienda iniciar proyectos orientados más hacia el enfoque laboral, donde los estudiantes recreen ambientes más productivos, que requieran de soluciones eficientes y rápidas.

4. Son muchas las aplicaciones que pueden ser mejoradas tomando en cuenta la temática de la seguridad, dentro de la Universidad. La información debe viajar por la red de manera segura y confiable. Este campo es de gran importancia para todo lo referente a la Web.

5. Se recomienda iniciar la implementación de módulos de seguridad cada vez más confiables y que aseguren la información, en todo tipo de aplicaciones que se dispongan a desarrollar.

BIBLIOGRAFIA

LIBROS Y ARTICULOS

Web Services Security Language (WS-Security)
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-security.asp

Cryptography and Security Services: Mechanisms and Applications
Autor Manuel Mogollón, Publicado en 2007

Securing Web services: practical usage of Standards and Specifications
Autor Panos Periorellis, Publicado en 2007

Enlaces y Seguridad
http://msdn.microsoft.com/es-es/library/ms731172.aspx

Aprende el ABC de la programación de Windows Comunicación Foundation
http://www.microsoft.com/spanish/msdn/articulos/archivo/041206/voices/LearnTheABCsOfP.mspx

TECNOLOGIA MAS EMPRESA IGUAL DESARROLLO. Disponible en www.degerencia.com/articulo

http://www.oracle.com/global/lad/technologies/soa/index.html

WEB SERVICES CONCEPTS, ARCHITECTURES AND APPLICATIONS. Gustavo Alonso, Fabio Casati, Harumi Kuno, Vijay Machiraju. Springer 2004

ENTERPRISE SERVICE ORIENTED ARCHITECTURES. CONCEPTS, CHALLENGES, RECOMMENDATIONS. James McGovern, Oliver Sims, Ashish Jain, Mark Little. Springer 2006.

http://www.uv.es/sto/cursos/seguridad.java/html/sjava-18.html

REFERENCIAS

[1] **OASIS,** *Organization for the advancement of Structured Information Standards*, consorcio internacional sin fines de lucro que orienta el desarrollo, la convergencia y la adopción de los estándares de comercio electrónico y Servicios Web, ellos deciden como y que trabajo se realiza mediante un proceso abierto y democrático.

[2] **File Transfer Protocol (FTP), Protocolo de Transferencia de Archivos** Protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP (Transmission Control Protocol), basado en la arquitectura cliente-servidor. Desde un equipo cliente se puede conectar a un servidor para descargar archivos desde él o para enviarle archivos, independientemente del sistema operativo utilizado en cada equipo.

El Servicio FTP es ofrecido por la capa de Aplicación del modelo de capas de red TCP/IP al usuario, utilizando normalmente el puerto de red 20 y el 21. Un problema básico de FTP es que está pensado para ofrecer la máxima velocidad en la conexión, pero no la máxima seguridad, ya que todo el intercambio de información, desde el login y password del usuario en el servidor hasta la transferencia de cualquier archivo, se realiza en texto plano sin ningún tipo de cifrado, con lo que un posible atacante puede capturar este tráfico, acceder al servidor, o apropiarse de los archivos transferidos. Para solucionar este problema son de gran utilidad aplicaciones como ***SCP y SFTP***, incluidas en el paquete ***SSH***, que permiten transferir archivos pero cifrando todo el tráfico.

[3] Abreviación de *World Wide Web Consortium*, es un consorcio internacional donde las organizaciones miembro, personal a tiempo completo y el público en general, trabajan conjuntamente para desarrollar estándares Web .La W3C fue fundada en 1994 por Tim Berners-Lee, el arquitecto de la World Wide Web. Como misión tiene el guiar la Web hacia su máximo potencial a través del desarrollo de protocolos y pautas que aseguren el crecimiento futuro de la Web.

[4] **Enterprise Application Integration (EAI),** o integración de Aplicaciones de Empresa se define como el uso de Software y principios de arquitectura de sistemas para integrar un conjunto de aplicaciones.

[5] **API**, **Interfaz de programación de aplicaciones** (del inglés *application programming interface*) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Usados generalmente en las bibliotecas.

[6] M**iddleware,** software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas. Funciona como una capa de abstracción de software distribuida, que se sitúa entre las capas de aplicaciones y las capas inferiores (sistema operativo y red). El middleware nos abstrae de la complejidad y heterogeneidad de las redes de comunicaciones subyacentes, así como de los sistemas operativos y lenguajes de programación, proporcionando una API para la fácil programación y manejo de aplicaciones distribuidas. Dependiendo del problema a resolver y de las funciones necesarias, serán útiles diferentes tipo de servicios de middleware.

[7] **Simple Mail Transfer Protocol (SMTP),** Protocolo de red basado en texto utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos (PDA's, teléfonos móviles, etc.). SMTP se basa en el modelo cliente-servidor, donde un cliente envía un mensaje a uno o varios receptores. La comunicación entre el cliente y el servidor consiste enteramente en líneas de texto compuestas por caracteres ASCII. El tamaño máximo permitido para estas líneas es de 1000 caracteres.

[10] Applied Cryptography Second Edition: protocols, algorithms and source code in C, Bruce Schneier, John Wiley & Sons, Inc., 1996

[11] FIPS PUB 180-1, Secure Hash Standard, http://www.itl.nist.gov/fipspubs/fip180-1.htm

[12] RFC 3174, Secure Hash Algorithm 1 (SHA-1), http://www.faqs.org/rfcs/rfc3174.html

[13] FIPS PUB 46-2, Data Encryption Standard (DES), http://www.itl.nist.gov/fipspubs/fip46-2.htm

[14] FIPS PUB 197, Advanced Encryption Standar (AES),

[15] Ten risks of PKI: What you're not being told about Public Key Infrastructure, Carl Ellison & Bruce Schneier, http://www.schneier.com/paper-pki.pdf

[16] RFC 4252, SSH Authentication Protocol, http://www.faqs.org/rfcs/rfc4252.html

[17] RFC 4253, SSH Transport Layer Protocol,
http://www.faqs.org/rfcs/rfc4253.html

[18] RFC 4254, SSH Connection Protocol, http://www.faqs.org/rfcs/rfc4254.html

[19] The SSL Protocol, V 3.0, http://wp.netscape.com/eng/ssl3/draft302.txt