

SEGURIDAD EN INTERNET

SSL

MIGUEL GARCIA FEAL

JOSE LUIS CHOUCIÑO FERREIRO

INDICE

1 – INTRODUCCIÓN.....	1
2 – SEGURIDAD EN INTERNET.....	2
3 – CERTIFICADOS DIGITALES.....	5
4 – VALIDEZ DE LOS CERTIFICADOS DIGITALES.....	8
5 – SECURE SOCKET LAYER.....	11
6 – PROTOCOLOS SSL.....	14
7 – VENTAJAS E INCONVENIENTES DE SSL.....	18
8 – SECURE ELECTRONIC TRANSATION.....	21
9 – ELEMENTOS DE SET.....	24
10 – CERTIFICADOS SET.....	26
11 – SERVIDORES SEGUROS.....	30
12 – COMO INSTALAR APACHE+SSL(+TOMCAT) EN LINUX.....	33
12.1 – PREPARACIÓN.....	33
12.2 – CINFIGURANDO EL SERVIDOR WEB APACHE	33
12.3 – INSTALANDO OPENSLL.....	33
12.4 – CINFUGURANDO MOD_SSL.....	33
12.5 – INSTALANDO EL SERVIDOR WEB APACHE.....	33
12.6 – COMPROBANDO QUE FUNCIONA.....	34

13 – COMO INSTALAR APACHE+SSL(+TOMCAT) EN WINDOWS.....	35
13.1 – PALABRAS PREVIAS.....	35
13.2 – INSTALANDO EL SERVIDOR WEB APACHE.....	35
13.3 – CREANDO NUESTRO CERTIFICADO.....	35
13.4 – CINFIGURANDO APACHE CON MOD_SSL.....	36
13.5 – COMPROBANDO QUE FUNCIONA.....	37
14 – EJEMPLO DE CONFIGURACION PARA SERVIDOR APACHE+PHP4+SSL+DOMINIOS VIRTUALES.....	38

1-INTRODUCCION

Seguro que a los desarrolladores de Internet, cuando idearon el sistema global de comunicación entre host de diferentes redes, nunca se les pasó por la cabeza que ese sistema iba a acabar siendo uno de los principales medios de intercambio de información de las décadas venideras. Y más seguro aún que cuando Tim Berners Lee sentó las bases del lenguaje HTML y de los documentos inter-enlazados mediante hipertexto jamás pudo imaginar que esto iba a ser la base no sólo de un sistema global de intercambio de datos y multimedia, si no la base de un floreciente mercado en el que empresas y bancos están haciendo su agosto (aunque haya muchas de ellas que quiebren).

Y es que la WWW ha sido tomada como base para una nueva forma de economía, basada en el e-commerce (o mejor dicho, en el comercio electrónico), que actualmente mueve miles de millones de pesetas-euros-dolares en todo el mundo.

Pero no todo es coser y cantar en este mundo cibereconómico. Internet fué diseñada y construida como un sistema de libre intercambio de información, por lo que aspectos como la seguridad en las transacciones realizadas no fueron implementados a su tiempo, con la consecuencia de que para hacerlo luego ha sido necesario apoyarse en una serie de sistemas y tecnologías adicionales.

Y es que una de las principales barreras que he encontrado el comercio electrónico ha sido precisamente la desconfianza de los usuarios frente a los sistemas de identificación, intercambio de datos personales y pago de productos a través de la grán red. Cosa por otro lado lógica, ya que, como hemos dicho antes, Internet es por propia naturaleza un sistema no-seguro.

Muchos han sido los intentos de las empresas y bancos por romper ese miedo innato de los usuarios, y muchos los sistemas que se han creado para ofrecerles medios seguros de comunicación y de pago (sobre todo, de pago), siempre con vistas a poder realizar esos negocios fabulosos que en un principio los visionarios de la red pronosticaban.

Los modernos sistemas de seguridad en transacciones a través de redes están basados en el uso de Infraestructuras de Clave Pública, basadas en un conjunto de elementos como Certificados Digitales, Criptografía simétrica y de clave pública, firmas digitales, Listas de Certificados Revocados, ect., que garantizan el cumplimiento de los 4 pilares de las comunicaciones seguras.

Vamos a estudiar en este pequeño manual qué son los certificados digitales y qué son Secure Socket Layer, SSL, y Secure Electronic Transaction, SET, con objeto de conocer un poco más cómo se produce el intercambio de datos sensibles, como el número de nuestra tarjeta de crédito y nuestros datos de cuentas, a través de la red de redes.

2 – EL PROBLEMA DE LA SEGURIDAD EN INTERNET

Como vimos en el manual sobre Criptología, para poder afirmar que una comunicación entre dos entidades es segura se deben cumplir cuatro requisitos principales:

1. **Autenticidad:** todas las entidades participantes en la transacción deben estar perfecta y debidamente identificadas antes de comenzar la misma. Debemos estar seguros de que la persona con la que nos comunicamos es realmente quién dice ser, ya que si no podemos estar facilitando datos íntimos y/o sensibles a una persona o entidad no deseada, que puede hacer con ellos luego lo que le venga en gana.

En las comunicaciones "normales" entre dos personas casi siempre se dispone de alguna forma de comprobación de la Autenticidad. Si hablamos en directo con alguien, sabemos quién es, y si no lo sabemos podemos poner límites a la información que le facilitamos. En una conversación telefónica podemos oír la voz de nuestro interlocutor, y si lo conocemos bien es muy difícil que otra persona se pueda hacer pasar por él.

Pero en nuestros viajes por la red no tenemos ninguna forma efectiva de saber con quién estamos comunicándonos. Podemos acudir a la página de una empresa, ver su dirección de dominio, ver en su página lo que nos dice, pero ¿cómo podemos saber que es realmente quien dice?. Imaginemos una situación extrema: un pirata informático sin escrúpulos crea una página igual en todo a la de nuestro banco y nos manda un correo diciéndonos que es el director de nuestra sucursal y que hay un problema con una de nuestras cuentas, ofreciéndonos un enlace para que entremos en las páginas del banco. Pero cuando pinchamos ese enlace resulta que nos lleva a su página falsificada, en la que nos pide que introduzcamos nuestras claves de acceso. En cuanto lo hagamos y le demos al botón de enviar, el pirata se hará con ellas, con todo lo que ello puede significar.

Lo ideal en este sentido sería que el cliente en una transacción de compra por Internet sólo debiera garantizar que es el legítimo propietario de la tarjeta de crédito que está usando en la misma, sin tener que hacer pública su identidad, por muchas leyes de protección de datos que estén vigentes.

La Autenticidad se consigue mediante el uso de los certificados y firmas digitales.

2. **Confidencialidad:** debemos estar seguros de que los datos que enviamos no pueden ser leídos por otra persona distinta del destinatario final deseado, o que si ocurre ésto, el espía no pueda conocer el mensaje enviado. O en su defecto, que cuando consiga obtener los datos éstos ya no le sirvan para nada. Es decir, debemos estar seguros de que ninguna persona ajena a la transacción puede tener acceso a los datos de la misma.

Imaginemos ahora que trabajamos en una empresa y deseamos enviar un correo al director general explicándole el fabuloso contrato que estamos a punto de firmar con un cliente. Si nuestro pirata de turno está a la escucha, puede conocer al momento todos los detalles del trato que vamos a realizar, pudiendo vender esa información a la competencia, lo que nos puede arruinar el negocio antes de hacerse realidad (¿qué impersión le causaría a nuestro cliente si recibiera información detallada de sus actividades con nosotros por medio de un correo anónimo?).

Lo ideal en este aspecto sería que las entidades implicadas en la transacción no llegaran a conocer más que los datos imprescindibles para realizar su función.

La confidencialidad se consigue en las transacciones electrónicas con el uso de la Criptografía.

3. **Integridad :** es necesario estar seguro de que los datos que enviamos llegan íntegros, sin modificaciones, a su destino final.

En este caso estamos realizando el pedido de un ordenador de 300.000 ptas a una tienda virtual, introducimos nuestro número de tarjeta de crédito y nuestra dirección de entrega del equipo. Pero nuestro simpático pirata está a la escucha, intercepta el envío, cambia los datos de la dirección por otros a su gusto y deja que continúe el envío. El resultado será que nuestro amigo disfrutará de un ordenador que hemos pagado nosotros.

La integridad se consigue combinando Criptografía, funciones hash y firmas digitales.

4. **No repudio** : debemos estar seguros de que una vez enviado un mensaje con datos importantes o sensibles el destinatario de los mismos no pueda negar el haberlos recibido. Y en una compra on-line debe garantizarse que una vez finalizada la misma ninguna de las partes que intervienen pueda negar haber participado en ella.

Vamos entonces a imaginar que nuestra empresa tiene que enviar un presupuesto antes de una fecha determinada, presupuesto que debe ser recogido por un empleado de otra empresa, y que éste olvida comunicar a sus superiores la recepción del presupuesto. Pasa el plazo y el contrato que esperábamos se lo dan a otra empresa, alegando que no han recibido a tiempo el presupuesto nuestro. Si no disponemos de un medio para atestiguar que el mensaje fué entregado en plazo, nos quedaremos sin contrato y sin poder reclamar.

Lo ideal sería que al finalizar la transacción quedara algo equivalente a un recibo de compra o factura firmado por todas las partes implicadas.

Independientemente de esto, la ley española fija que todo cliente que participe en una compra a distancia (por Internet, por catálogo, etc.) tiene derecho a negar su participación en ella.

El no repudio se consigue mediante los certificados y la firma digital.

* Bien, amigos. Estas son las condiciones que debe cumplir una comunicación por Internet para poder considerarse como segura, y como hemos visto, esta seguridad la podemos conseguir en todos sus aspectos usando el equipo formado por Criptografía y firma digital (ver manual en HTMLWeb) y los certificados digitales. Todos estos ingredientes se combinan en diferentes tecnologías, que veremos a continuación.

3 – CERTIFICADOS DIGITALES

Para solucionar el problema de la Autenticación en las transacciones por Internet se buscó algún sistema identificativo único de una entidad o persona. Ya existían los sistemas criptográficos de clave asimétrica, mediante los cuales una persona disponía de dos claves, una pública, al alcance de todos, y otra privada, sólo conocida por el propietario. Cuando deseamos enviar un mensaje confidencial a otra persona, basta pues con cifrarlo con su clave pública, y así estaremos seguros de que sólo el destinatario correcto podrá leer el mensaje en claro.

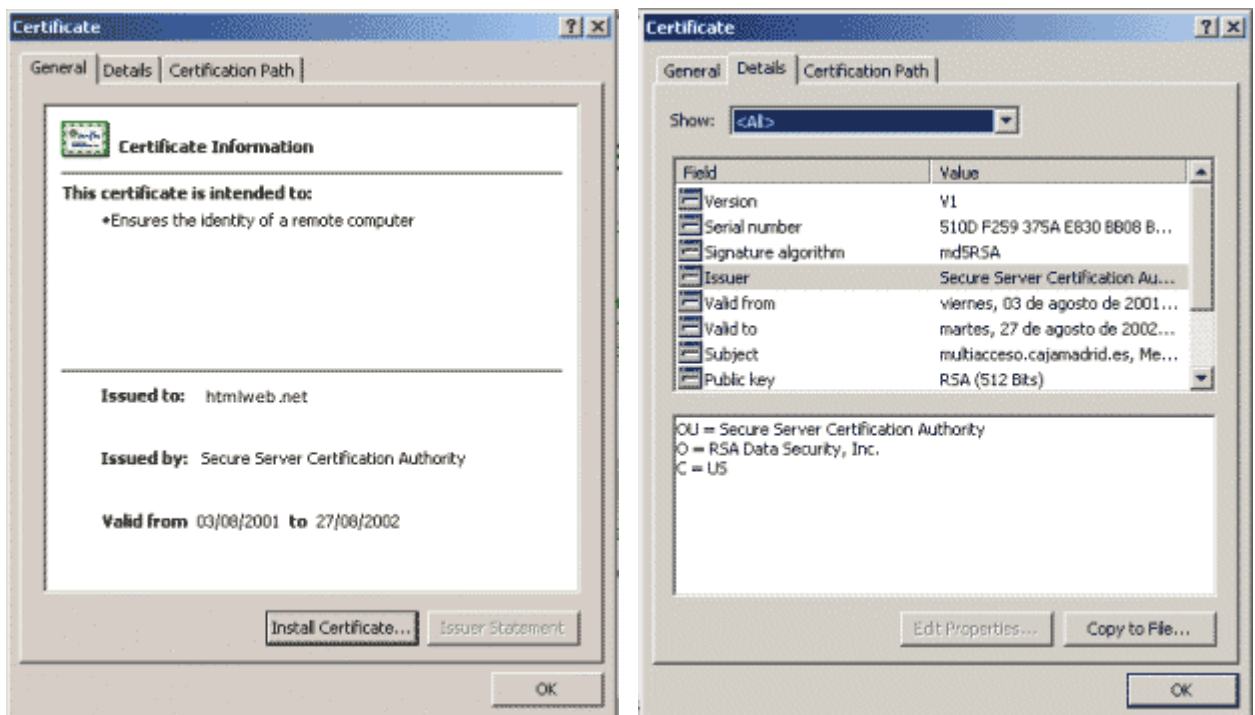
El problema era estar seguro de que efectivamente la clave pública que nos envían sea de la persona correcta, y no de un suplantador. Entonces se pensó en implementar una especie de documento de identidad electrónica que identificara sin dudas a su emisor.

La solución a este problema la trajo la aparición de los **Certificados Digitales** o **Certificados Electrónicos**, documentos electrónicos basados en la criptografía de clave pública y en el sistema de firmas digitales (ver manual de Criptografía para más datos). La misión principal de un Certificado Digital es garantizar con toda confianza el vínculo existente entre una persona, entidad o servidor web con una pareja de claves correspondientes a un sistema criptográfico de clave pública.

Un Certificado Digital es un documento electrónico que contiene datos identificativos de una persona o entidad (empresa, servidor web, etc.) y la llave pública de la misma, haciéndose responsable de la autenticidad de los datos que figuran en el certificado otra persona o entidad de confianza, denominada **Autoridad Certificadora**. Las principales Autoridades Certificadoras actuales son Verisign (filial de RSA Data Security Inc.) y Thawte.

El sistema es análogo a otros de uso común, como en D.N.I. español, en el que una autoridad de confianza (el estado o la policía) atestigua que la persona portadora de dicho documento es quién dice ser.

El formato de los Certificados Digitales es estándar, siendo X.509 v3 el recomendado por la Unión Internacional de Comunicaciones (ITU) y el que está en vigor en la actualidad. El aspecto de los certificados X.509 v3 es el siguiente:



Los datos que figuran generalmente en un certificado son:

1. Versión: versión del estándar X.509, generalmente la 3, que es la más actual.
2. Número de serie: número identificador del certificado, único para cada certificado expedido por una AC determinada.
3. Algoritmo de firma: algoritmo criptográfico usado para la firma digital.
4. Autoridad Certificadora: datos sobre la autoridad que expide el certificado.
5. Fechas de inicio y de fin de validez del certificado. Definen el periodo de validez del mismo, que generalmente es de un año.
6. Propietario: persona o entidad vinculada al certificado. Dentro de este apartado se usan una serie de abreviaturas para establecer datos de identidad. Un ejemplo sería:

Field	Value
Serial number	510D F259 375A E830 BB08 B...
Signature algorithm	md5RSA
Issuer	Secure Server Certification Au...
Valid from	viernes, 03 de agosto de 2001...
Valid to	martes, 27 de agosto de 2002...
Subject	htmlweb.net
Public key	RSA (512 Bits)
Thumbprint algorithm	sha1

CN = htmlweb.net
OU = Member, VeriSign Trust Network
OU = Authenticated by Telefonica S.A.
OU = Terms of use at www.ace.es/rpa (c) 01
OU = Educación
O = HTMLWeb
L = Madrid
S = Madrid
C = ES

CN	nombre común del usuario
OU	información varia
O	organización
L	ciudad
S	estado (provincia)
C	país
E	correo electrónico
UID	ID de usuario

7. Llave pública: representación de la llave pública vinculada a la persona o entidad (en hexadecimal), junto con el algoritmo criptográfico para el que es aplicable.
8. Algoritmo usado para la misma para obtener la firma digital de la Autoridad Certificadora.
9. Firma de la Autoridad Certificadora, que asegura la autenticidad del mismo.
10. Información adicional, como tipo de certificado, etc.

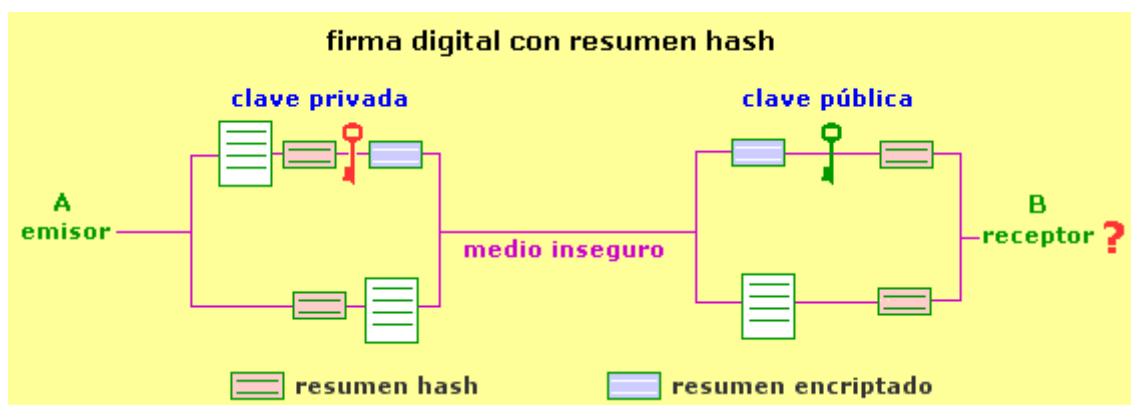
El problema que se plantea ahora es: si la Autoridad Certificadora avala los datos del certificado ¿Quién avala a la autoridad Certificadora?. Para solventar esto se han creado una serie de entidades

autorizadas a emitir certificados, de tal forma que éstas a su vez son avaladas por otras entidades de mayor confianza, hasta llegar a la cabeza de la jerarquía, en la que figuran unas pocas entidades de reconocido prestigio y confianza, como Verisign, que se autofirman su certificado.

Cada certificado emitido por una AC debe estar firmado por una AC de mayor grado en el esquema jerárquico de autoridades certificadoras, formándose así una cadena de certificados, en los que unas AC se avalan a otras hasta llegar a la AC superior, que se avala a sí misma. La jerarquía de firmas y la cadena con ella formada están contempladas en el estándar X.509 v3, que indica la forma correcta de realizar estas cadenas de certificaciones.

El certificado Digital vincula pues indisolublemente a una persona o entidad con una llave pública, y mediante el sistema de firma digital se asegura que el certificado que recibimos es realmente de la persona que consta en el mismo. El sistema de firma digital liga un documento digital con una clave de cifrado.

El procedimiento de **firma digital** lo que hace es obtener un resumen de un documento o de un texto aleatorio y cifrarlo con llave privada del propietario del certificado. Cuando nos llega un certificado, y su firma digital asociada, tan sólo debemos obtener nosotros el resumen el mismo, descifrar la firma con la llave pública del remitente y comprobar que ambos resúmenes coinciden, lo que nos hace estar totalmente seguros de la autenticidad del certificado. Se firma un resumen del documento y no el documento mismo para evitar ataques contra el sistema de cifrado RSA (por ejemplo, encriptar un documento especialmente concebido por un pirata, con lo que éste podría llegar a obtener la llave privada) y para no hacer el proceso demasiado lento.



Para obtener el resumen del documento se utilizan las **funciones hash** o de resumen, algoritmos criptográficos muy rápidos, de uso público e irreversibles (de un sólo sentido). Son funciones de dispersión que no usan ninguna clave, y que transforman el mensaje original en una cadena de dígitos de longitud fija (generalmente de entre 16 y 128 bits).

Los procesos de validación de certificados, obtención de resúmenes, descifrados y comprobación de coincidencia se realizan por el software adecuado del navegador web o programa de seguridad particular de forma transparente al usuario, por lo que éste será informado sólo en el caso de que el certificado no sea válido.

4 – VALIDEZ DE LOS CERTIFICADOS DIGITALES

Los certificados, debido a su propia naturaleza y al papel que desempeñan, no son documentos imperecederos, al igual que sucede con el resto de documentos de autenticación de otros tipos.

En primer lugar, al estar basados en el uso de claves no conviene que sean válidos por periodos de tiempo largos, ya que uno de los principales problemas del manejo de claves es que cuanto más vida tienen más fácil es que alguien extraño se apodere de ellas. Además, con el paso del tiempo los equipos informáticos van teniendo cada vez más poder de cálculo, facilitando con ello la labor de los criptoanalistas, por lo que es conveniente que cada cierto tiempo se vaya aumentando el tamaño de las claves criptográficas. Por este motivo los Certificados Digitales tienen estipulado un periodo de validez, que suele ser de un año.

En segundo lugar, es posible que un certificado convenga anularlo en un momento dado, bien porque se crea que las claves estén comprometidas, bien porque la persona o entidad propietaria haya caído en quiebra o delito. Es por esto que existe la posibilidad de revocar o anular un certificado, y esta revocación puede llevarla a cabo el propietario del mismo, la Autoridad Certificadora o las autoridades judiciales.

Para llevar un control de los certificados revocados (no válidos) las Autoridades de Certificación han implementado unos servidores especiales que contienen bases de datos en las que figuran los certificados anulados, que se conocen con el nombre de **Lista de Certificados Revocados, CRL**. Un CRL es pues un archivo, firmado por la Autoridad Certificadora, que contiene la fecha de emisión del mismo y una lista de certificados revocados, figurando para cada uno de ellos su número de identificación y la fecha en que ha sido revocado.

Cuando nuestro software de seguridad recibe un Certificado Digital de otra persona o entidad comprueba antes de darlo por bueno si dicho certificado se encuentra en la lista más actualizada de certificados revocados. Si está en la lista, el certificado será rechazado.

Ahora bien, imaginemos que recibimos un certificado como medio de autenticación en una transacción, nuestro software comprueba que no está revocado en la última CRL y lo da por válido, pero resulta que al día siguiente aparece como revocado en la CRL nueva. En estos casos deberemos poder demostrar de algún modo que hemos recibido el certificado antes de que se produjera la actualización.

Para solucionar este tipo de situaciones existen los documentos digitales denominados **recibos**. Un recibo es un documento firmado digitalmente por una persona o entidad de confianza, llamada **Autoridad de Oficialía de Partes**, que añade la fecha actual a los documentos que recibe para su certificación, firmando luego el resultado con su llave privada. De esta forma los usuarios disponen de un documento que atestigüe la hora y fecha exacta en la que envía o recibe un Certificado Digital u otro documento electrónico cualquiera.

Resumiendo, mediante la consulta a una Lista de Certificados Revocados y un recibo de una Autoridad de Oficialía de partes disponemos de pruebas suficientes para considerar cualquier transacción realizada en base a Certificados Digitales como segura (por lo menos en el sentido de Autenticación).

El uso de un CRL en un proceso de Autenticación presenta varios problemas adicionales. En primer lugar sólo podemos considerarlo válido cuando la fecha del mismo es igual o posterior a la que queremos usar como referencia en la validez del documento, y en segundo lugar, también puede resultar inadecuado en aquellas operaciones que exijan una velocidad alta en la transacción, sobre todo si el CRL a consultar tiene un tamaño muy grande.

La solución a estos problemas la dan los **Servicios de Directorios o de Consulta de Certificados**, servicios ofrecidos por personas o entidades de confianza aceptada, por el que al recibir una petición de validez de un certificado responde al instante si en esa fecha y hora concreta el mismo es válido o si por el contrario está revocado, en cuyo caso proporcionará también la fecha UTC de revocación. Para dar validez a la respuesta, el Servicio de Directorios firma con su llave privada la misma, con lo que el usuario estará seguro de la Autenticidad de la respuesta recibida.

EMISIÓN DE CERTIFICADOS DIGITALES

Los Certificados Digitales, como ya hemos dicho, son emitidos por las Autoridades de Certificación, entidades consideradas de confianza probada, como Verisign, Cybertrust o Nortel. Al hacerse responsables estas entidades de los certificados que emiten, dando fe de la relación existente entre los datos que figuran en un certificado y la persona o entidad que lo solicita, una de las tareas más importantes de las mismas es ejercer un control estricto sobre la exactitud y veracidad de los datos incorporados en el certificado.

Para solicitar un certificado a una AC la persona o entidad interesada debe cumplir unos procedimientos previos, confeccionando un documento, denominado **Requerimiento de Certificación**, en el que deben figurar los datos representativos del solicitante (nombre personal o de empresa, domicilio personal o social, dominio asociado a la empresa y al servidor seguro, etc.) y su llave pública. También debe manifestar su voluntad de aceptar dicha llave pública y demostrar que es el propietario real de la llave privada asociada, mediante el firmado digital de un mensaje.

La presentación de todos estos datos ante la Autoridad Certificadora puede acarrear problemas, al estar éstas normalmente muy distantes de los solicitantes. Para solventar esto se han creado unas entidades intermedias, conocidas como **Autoridades Registradoras**, autorizadas por las AC, y cuya misión es comprobar la validez de los datos presentados en el Requerimiento de Certificación. Una vez comprobados, las AR envía el OK a las AC, que emiten el correspondiente Certificado Digital.

Para que se pueda obtener con facilidad el Certificado Digital de cualquier persona o entidad las Autoridades de Certificación disponen de servidores de acceso público que realizan la función de depósito de certificados, en los que se puede buscar el deseado y descargarlo a nuestro ordenador. Es ésta una forma más segura que la de usar directamente un certificado recibido por correo o descargado de una página web, ya que la Autoridad de Certificación responsable del servidor es la encargada de verificar constantemente la validez y autenticidad de los certificados que distribuye.

Además de las Autoridades de Certificación reconocidas existen otras entidades que también pueden expedir certificados. Este es el caso de entidades gubernamentales (como el Servicio Postal de EEUU) y ciertas corporaciones empresariales que compran un servicio de certificación a un vendedor que haya sido a su vez certificado por una AC. Estos certificados se suelen usar para empleados de la propia compañía que deben hacer negocios para ella. Se espera que en el futuro este tipo de certificados adquiera cada vez mayor importancia.

TIPOS DE CERTIFICADOS

Dependiendo del uso que se vaya a dar al certificado y de qué persona o entidad lo solicita, las Autoridades Certificadoras han dividido los certificados en varios tipos. Del tipo de certificado a emitir van a depender las medidas de comprobación de los datos y el precio del mismo.

Los certificados, según las comprobaciones de los datos que se realizan, se dividen en cuatro clases:

* **Certificados de Clase 1:** corresponde a los certificados más fáciles de obtener e involucran pocas verificaciones de los datos que figuran en él: sólo el nombre y la dirección de correo electrónico del titular.

* **Certificados de Clase 2:** en los que la Autoridad Certificadora comprueba además el permiso de conducir, el número de la Seguridad Social y la fecha de nacimiento.

* **Certificados de Clase 3:** en la que se añaden a las comprobaciones de la Clase 2 la verificación de crédito de la persona o empresa mediante un servicio como Equifax.

* **Certificados de Clase 4:** que a todas las comprobaciones anteriores suma la verificación del cargo o la posición de una persona dentro de una organización (todavía no formalizados los requerimientos; está en estudio).

Desde el punto de vista de la finalidad, los certificados electrónicos se dividen en:

1. **Certificados SSL para cliente:** usados para identificar y autenticar a clientes ante servidores en comunicaciones mediante el protocolo Secure Socket Layer, y se expiden normalmente a una persona física, bien un particular, bien un empleado de una empresa.

2. **Certificados SSL para servidor:** usados para identificar a un servidor ante un cliente en comunicaciones mediante el protocolo Secure Socket Layer, y se expiden generalmente a nombre de la empresa propietaria del servidor seguro o del servicio que éste va a ofrecer, vinculando también el dominio por el que se debe acceder al servidor. La presencia de éste certificado es condición imprescindible para establecer comunicaciones seguras SSL.

3. **Certificados S/MIME :** usados para servicios de correo electrónico firmado y cifrado, que se expiden generalmente a una persona física. El mensaje lo firma digitalmente el remitente, lo que proporciona Autenticación, Integridad y No Rechazo. También se puede cifrar el mensaje con la llave pública del destinatario, lo que proporciona Confidencialidad al envío.

4. **Certificados de firma de objetos:** usados para identificar al autor de ficheros o porciones de código en cualquier lenguaje de programación que se deba ejecutar en red (Java, JavaScript, CGI, etc). Cuando un código de éste tipo puede resultar peligroso para el sistema del usuario, el navegador lanza un aviso de alerta, en el que figurará si existe certificado que avale al código, con lo que el usuario puede elegir si confía en el autor, dejando que se ejecute el código, o si por el contrario no confía en él, con lo que el código será rechazado.

5. **Certificados para AC:** que identifican a las propias Autoridades Certificadoras, y es usado por el software cliente para determinar si pueden confiar en un certificado cualquiera, accediendo al certificado de la AC y comprobando que ésta es de confianza.

Toda persona o entidad que desée obtener un certificado debe pagar una cuota a las Autoridades de Certificación, cuota que irá en función de la clase del certificado y del uso que se le vaya a dar al mismo (ambas están relacionadas). A mayor nivel de comprobación de datos (clase mayor), más costará el certificado.

5 – SECURE SOCKET LAYER

Como vimos al principio del tema, toda transacción segura por la red debe contemplar los aspectos de Autenticidad, Integridad, Confidencialidad y No Repudio. Son varios los sistemas y tecnologías que se han desarrollado para intentar implementar estos aspectos en las transacciones electrónicas, siendo sin duda SSL el más conocido y usado en la actualidad. SSL permite la Confidencialidad y la Autenticación en las transacciones por Internet, siendo usado principalmente en aquellas transacciones en la que se intercambian datos sensibles, como números de tarjetas de crédito o contraseñas de acceso a sistemas privados. SSL es una de las formas base para la implementación de soluciones PKI (Infraestructuras de Clave Pública).

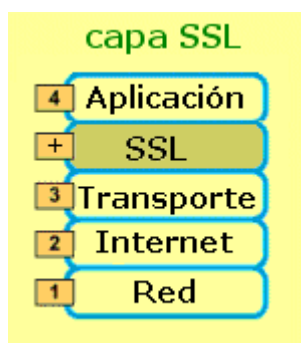
Secure Socket Layer es un sistema de protocolos de carácter general diseñado en 1994 por la empresa Netscape Communications Corporation, y está basado en la aplicación conjunta de Criptografía Simétrica, Criptografía Asimétrica (de llave pública), certificados digitales y firmas digitales para conseguir un canal o medio seguro de comunicación a través de Internet. De los sistemas criptográficos simétricos, motor principal de la encriptación de datos transferidos en la comunicación, se aprovecha la rapidez de operación, mientras que los sistemas asimétricos se usan para el intercambio seguro de las claves simétricas, consiguiendo con ello resolver el problema de la Confidencialidad en la transmisión de datos.

SSL implementa un protocolo de negociación para establecer una comunicación segura a nivel de socket (nombre de máquina más puerto), de forma transparente al usuario y a las aplicaciones que lo usan.

Actualmente es el estándar de comunicación segura en los navegadores web más importantes (protocolo HTTP), como Netscape Navigator e Internet Explorer, y se espera que pronto se saquen versiones para otros otros protocolos de la capa de Aplicación (correo, FTP, etc.).

La identidad del servidor web seguro (y a veces también del usuario cliente) se consigue mediante el Certificado Digital correspondiente, del que se comprueba su validez antes de iniciar el intercambio de datos sensibles (Autenticación), mientras que de la seguridad de Integridad de los datos intercambiados se encarga la Firma Digital mediante funciones hash y la comprobación de resúmenes de todos los datos enviados y recibidos.

Desde el punto de vista de su implementación en los modelos de referencia OSI y TCP/IP, SSL se introduce como una especie de nivel o capa adicional, situada entre la capa de Aplicación y la capa de Transporte, sustituyendo los sockets del sistema operativo, lo que hace que sea independiente de la aplicación que lo utilice, y se implementa generalmente en el puerto 443. (NOTA: Los puertos son las interfaces que hay entre las aplicaciones y la pila de protocolos TCP/IP del sistema operativo).



SSL proporciona servicios de seguridad a la pila de protocolos, encriptando los datos salientes de la capa de Aplicación antes de que estos sean segmentados en la capa de Transporte y encapsulados y enviados por las capas inferiores. Es más, también puede aplicar algoritmos de compresión a los datos a enviar y fragmentar los bloques de tamaño mayor a 2^{14} bytes, volviéndolos a reensamblarlos en el receptor.

La versión más actual de SSL es la 3.0. que usa los algoritmos simétricos de encriptación DES, TRIPLE DES, RC2, RC4 e IDEA, el asimétrico RSA, la función hash MD5 y el algoritmo de firma SHA-1.

Los algoritmos, longitudes de clave y funciones hash de resumen usados en SSL dependen del nivel de seguridad que se busque o se permita, siendo los más habituales los siguientes:

* **RSA + Triple DES de 168 bits + SHA-1**: soportado por las versiones 2.0 y 3.0 de SSL, es uno de los conjuntos más fuertes en cuanto a seguridad, ya que son posibles $3.7 * 10^{50}$ claves simétricas diferentes, por lo que es muy difícil de romper. Por ahora sólo está permitido su uso en Estados Unidos, aplicándose sobre todo en transacciones bancarias.

* **RSA + RC4 de 128 bits + MD5**: soportado por las versiones 2.0 y 3.0 de SSL, permite $3.4 * 10^{38}$ claves simétricas diferentes que, aunque es un número inferior que el del caso anterior, da la misma fortaleza al sistema. Análogamente, en teoría sólo se permite su uso comercial en Estados Unidos, aunque actualmente ya es posible su implementación en los navegadores más comunes, siendo usado por organismos gubernamentales, grandes empresas y entidades bancarias.

* **RSA + RC2 de 128 bits + MD5**: soportado sólo por SSL 2.0, permite $3.4 * 10^{38}$ claves simétricas diferentes, y es de fortaleza similar a los anteriores, aunque es más lento a la hora de operar. Sólo se permite su uso comercial en Estados Unidos, aunque actualmente ya es posible su implementación en los navegadores más comunes.

* **RSA + DES de 56 bits + SHA-1**: soportado por las versiones 2.0 y 3.0 de SSL, aunque es el caso de la versión 2.0 se suele usar MD5 en vez de SHA-1. Es un sistema menos seguro que los anteriores, permitiendo $7.2 * 10^{16}$ claves simétricas diferentes, y es el que suelen traer por defecto los navegadores web en la actualidad (en realidad son 48 bits para clave y 8 para comprobación de errores).

* **RSA + RC4 de 40 bits + MD5**: soportado por las versiones 2.0 y 3.0 de SSL, ha sido el sistema más común permitido para exportaciones fuera de Estados Unidos. Permite aproximadamente $1.1 * 10^{12}$ claves simétricas diferentes, y una velocidad de proceso muy elevada, aunque su seguridad es ya cuestionable con las técnicas de Criptoanálisis actuales.

* **RSA + RC2 de 40 bits + MD5**: en todo análogo al sistema anterior, aunque de velocidad de proceso bastante inferior.

* **Sólo MD5**: usado sólomente para autenticar mensajes y descubrir ataques a la integridad de los mismos. Se usa cuando el navegador cliente y el servidor no tienen ningún sistema SSL común, lo que hace imposible el establecimiento de una comunicación cifrada. No es soportado por SSL 2.0, pero si por la versión 3.0.

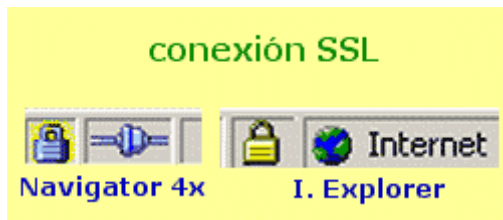
La clave de encriptación simétrica es única y diferente para cada sesión, por lo que si la comunicación falla y se debe establecer una nueva sesión SSL, la contraseña simétrica se generará de nuevo.

SSL proporciona cifrado de alto nivel de los datos intercambiados (se cifran incluso las cabeceras HTTP), autenticación del servidor (y si es necesario también del cliente) e integridad de los datos recibidos.

Durante el proceso de comunicación segura SSL existen dos estados fundamentales, el **estado de sesión** y el **estado de conexión**. A cada sesión se le asigna un número identificador arbitrario, elegido por el servidor, un método de compresión de datos, una serie de algoritmos de encriptación y funciones hash, una clave secreta maestra de 48 bytes y un flag de nuevas conexiones, que indica si desde la sesión actual se pueden establecer nuevas conexiones. Cada conexión incluye un número secreto para el cliente y otro para el servidor, usados para calcular los MAC de sus mensajes, una clave secreta de encriptación particular para el cliente y otra para el servidor, unos vectores iniciales en el caso de cifrado de datos en bloque y unos números de secuencia asociados a cada mensaje.

¿Cómo podemos saber si una conexión se está realizando mediante SSL?. Generalmente los navegadores disponen de un icono que lo indica, generalmente un candado en la parte inferior de la ventana. Si el candado está abierto se trata de una conexión normal, y si está cerrado de una

conexión segura. Si hacemos doble click sobre el candado cerrado nos aparecerá el Certificado Digital del servidor web seguro.



Además, las páginas que proceden de un servidor SSL vienen implementadas mediante protocolo HTTP seguro, por lo que su dirección, que veremos en la barra de direcciones del navegador, empezará siempre por https, como por ejemplo:

`https://www.htmlweb.net`

Por último, cuando estamos en una conexión segura podemos ver el certificado del servidor acudiendo al menú "Archivo" del navegador y pinchando en "Propiedades". En la parte inferior tenemos una opción "Certificados", que nos mostrará el del servidor actual.

Vamos a ver a continuación de forma detallada el proceso completo de trabajo de SSL.

6 – PROTOCOLOS SECURE SOCKET LAYER

Para establecer una comunicación SSL es necesario que previamente el cliente y el servidor realicen un proceso de reconocimiento mutuo y de petición de conexión que, al igual que en otros tipos de comunicaciones, recibe el nombre de apretón de manos o Handshake, que en este caso está controlado por el **Potocolo SSL Handshake**, que se encarga de establecer, mantener y finalizar las conexiones SSL. Durante el mismo se negocian los parámetros generales de la sesión y los particulares de cada conexión.

Concretamente, y de forma general, el protocolo comienza con el saludo del cliente al servidor, conocido como **Client Hello**, por el que se informa al servidor de que se desea establecer una comunicación segura con él. SSL soporta solicitudes de conexión por puertos diferentes al utilizado normalmente para este servicio. Junto con este saludo inicial, el cliente envía al servidor información de la versión de SSL que tiene implementada, de los algoritmos de encriptación que soporta, las longitudes de clave máximas que admite para cada uno de ellos y las funciones hash que puede utilizar. También se le solicita al servidor el envío de su Certificado Digital X.509 v3, con objeto de verificar el cliente la identidad del mismo y recoger su clave pública. En este momento se asigna un identificador a la sesión y se hace constar la hora y fecha de la misma.

Como medida adicional, el cliente envía asimismo una clave numérica aleatoria, para que se pueda establecer una comunicación segura mediante otros protocolos o algoritmos en el caso de que el servidor web no poséa un Certificado Digital.

En este paso no se intercambia en ningún momento información sensible, tan sólo información necesaria para establecer la comunicación segura.

A continuación, el servidor SSL responde al cliente en el proceso que se conoce con el nombre de **Server Hello**, enviándole su Certificado Digital (con su llave pública) e informándole de su versión de SSL, de los algoritmos y longitudes de clave que soporta.

Generalmente se obtiene el conjunto de algoritmos, longitudes de clave y funciones hash soportados por ambos, eligiéndose entonces los más fuertes. Si no hay acuerdo con los algoritmos a usar se envía un mensaje de error.

A veces, y si la comunicación posterior así lo exige, el servidor solicita al cliente su Certificado Digital, en el mensaje llamado **CertificateRequest**. Esto sólo suele ocurrir en SSL cuando los datos a transferir sean especialmente sensibles y precisen la previa autenticación del cliente. Si es el caso, el cliente debe contestar al servidor mediante el mensaje **CertificateVerify**, enviándole entonces su certificado.

En este momento el cliente verifica la validez del Certificado Digital del servidor, descriptando el resumen del mismo y comprobando su corrección, verificando que ha sido emitido por una Autoridad Certificadora de confianza, que esté correctamente firmado por ella y que el certificado no esté revocado. También se comprueba que la fecha actual está dentro del rango de fechas válidas para el certificado y que el dominio (URL) que aparece en el certificado se corresponde con el que se está intentando establecer la comunicación segura. Si alguna de estas validaciones falla, el navegador cliente rechazará la comunicación, dándola por finalizada e informando al usuario del motivo del rechazo.

En caso de que el servidor no tenga un Certificado X.509 v3 se puede utilizar un mensaje **ServerKeyExchange** para enviar la clave pública sin certificado, en cuyo caso queda en manos del cliente la elección de si acepta la llave o no, lo que finalizaría el proceso.

Como medida adicional de seguridad, el cliente genera una clave aleatoria temporal y se la envía al servidor, que debe devolvérsela cifrada con su clave privada. El cliente la descifra con la llave pública y comprueba la coincidencia, con lo que está totalmente seguro de que el servidor es quién dice ser. Y un proceso análogo a éste, pero en sentido inverso, se requiere si es necesaria la autenticación del usuario ante el servidor.

Si todo está correcto el cliente genera un número aleatorio que va a servir para calcular una clave de sesión correspondiente al algoritmo de encriptación simétrico negociado antes, conocida con el nombre de **clave maestra**, que es enviada al servidor de forma segura encriptándola asimétricamente con la llave pública del mismo que aparece en el Certificado Digital. Esta clave maestra se usará para generar todas las claves y números secretos utilizados en SSL.

Con esto servidor y cliente se han identificado y tienen en su poder todos los componentes necesarios para empezar a transmitir información cifrada simétricamente.

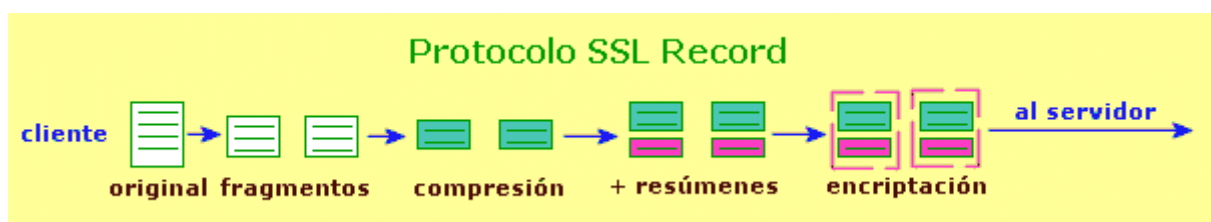
Se pasa entonces el control al subprotocolo Change Cipher Spec (ver más abajo), iniciándose la conexión segura.

Así y todo, para que empiecen las transmisiones de datos protegidos se requiere otra verificación previa, denominada **Finished**, consistente en que cliente y servidor se envían uno al otro una copia de todas las transacciones llevadas a cabo hasta el momento, encriptándola con la llave simétrica común. Al recibir esta copia, cada host la descripta y la compara con el registro propio de las transacciones. Si las transacciones de los dos host coinciden significa que los datos enviados y recibidos durante todo el proceso no han sido modificados por un tercero. Se termina entonces la fase Handshake.

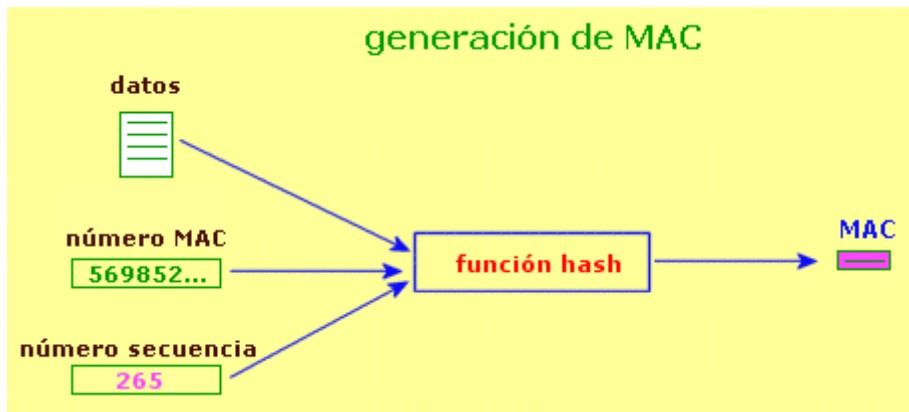
Para empezar a transmitir datos cifrados es necesario que cliente y servidor se pongan de acuerdo respecto a la forma común de encapsular los datos que se van a intercambiar, es decir, qué formato de datos se va a usar en la transmisión cifrada. Esto se realiza mediante el **Protocolo SSL Record** (Protocolo de Registro SSL), que establece tres componentes para la porción de datos del protocolo:

1. MAC-DATA: código de autenticación del mensaje.
2. ACTUAL-DATA: datos de aplicación a transmitir.
3. PADDING-DATA: datos requeridos para rellenar el mensaje cuando se usa un sistema de cifrado en bloque.

El Protocolo de Registro es el encargado de la seguridad en el intercambio los datos que le llegan desde desde las aplicaciones superiores, usando para ello los parámetros de encriptación y resumen negociados previamente mediante el protocolo SSL Handshake. Sus principales misiones son:



- La fragmentación de los mensajes mayores de 2^{14} bytes en bloques más pequeños.
- La compresión de los bloques obtenidos mediante el algoritmo de compresión negociado anteriormente.
- La autenticación y la integridad de los datos recibidos mediante el resumen de cada mensaje recibido concatenado con un número de de secuencia y un número secreto establecidos en el estado de conexión. El resultado de esta concatenación se denomina **MAC**, y se añade al mensaje. Con esta base, la autenticación se comprueba mediante el número secreto, compartido por el cliente y el servidor, y mediante el número de secuencia, que viaja siempre encriptado. La integridad se comprueba mediante la función hash negociada.



- La confidencialidad se asegura encriptando los bloques y sus resúmenes mediante el algoritmo simétrico y la clave correspondiente negociadas en la fase Handshake. Existen dos tipos posibles de encriptación:

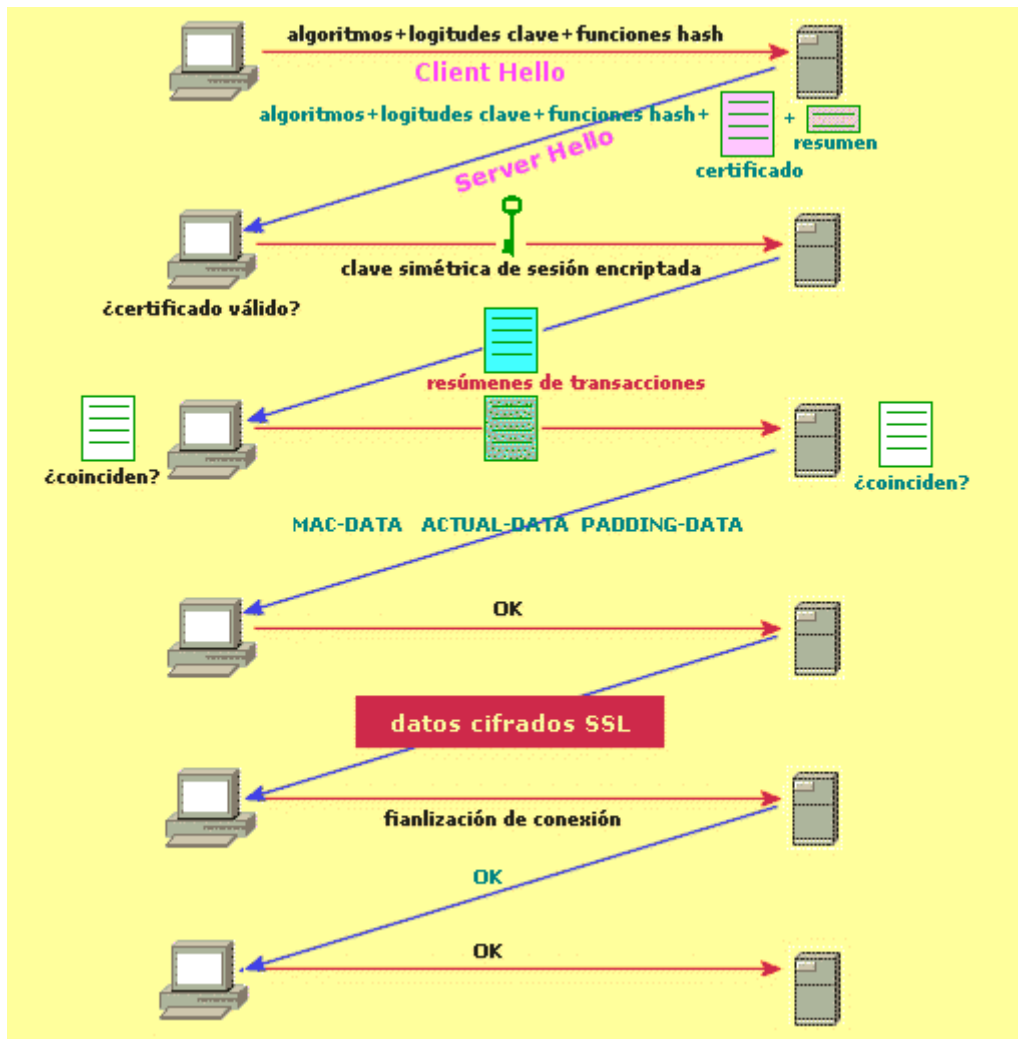
a. **Cifrado en bloque:** se cifran los datos en bloques de 64 bits. Si el mensaje no es múltiplo de 64 bits se le añaden los bits de relleno necesarios para obtener un número entero de bloques completos, indicándose la adición en el formato del mensaje. Este método de cifrado se conoce con el nombre de **Cipher Block Chaining, CBC**, y precisa un vector inicial, que habrá sido negociado previamente en la fase Handshake. Como algoritmos de cifrado se usan RC2 y DES.

b. **Cifrado Stream:** o de flujo, en el que se encriptan los datos realizando una operación lógica OR-Exclusiva entre los bytes y un generador pseudoaleatorio usando el algoritmo RC4.

Trás todos estos requisitos, el canal seguro está listo para empezar la transmisión de datos de forma segura. Cuando el cliente o el servidor deséan transmitir algún mensaje al otro se genera automáticamente un resumen del mismo mediante la función has acordada, se encriptan mensaje y resumen con la clave simétrica acordada y se envían los datos. Cuando el destinatario los recibe, desencripta todo, vuelve a obtener el resumen a partir del original y lo compara con el recibido. Si coinciden hay seguridad de que la comunicación segura se ha producido satisfactorimente, sin intromisiones externas. Si no coinciden, se pone en conocimiento del otro host, y si es preciso se suspende la conexión SSL. Cada uno de los mensajes enviados por cliente o servidor sufre este proceso de verificación.

Por último, cuando la transferencia de mensajes ha finalizado y se desea cerrar la comunicación segura, generalmente porque el cliente así lo deséa, la aplicación cliente (el navegador web, p.e.) lanza una ventana de aviso de que se va a cerrar la comunicación SSL, y si es aceptada por el usuario, se sale de la misma y se regresa a una comunicación normal, finalizando el proceso SSL.

Una visión gráfica de todo el proceso



SSL actúa computacionalmente como una máquina de estados: durante el intercambio de datos hay en todo momento un estado de escritura activo y otro pendiente y lo mismo ocurre respecto a la lectura de datos, realizándose el cambio de estados mediante un subprotocolo especial del Handshake denominado **Change Cipher Spec**.

SSL Handshake posee además otro subprotocolo específico, denominado **Alerta**, que se encarga de avisar de los problemas que ocurren durante la conexión, y que pueden llevar a la finalización brusca de la sesión.

7 – IMPLEMENTACION DEL PROTOCOLO SSL

Por la parte del cliente, SSL viene implementado por defecto en los navegadores Internet Explorer y Netscape Navigator, lo que permite a cualquier usuario con uno de estos navegadores poder realizar compras por Internet de forma segura sin tener que conocer el sistema a fondo ni preocuparse de instalar programas adicionales (por lo menos autenticando al servidor web y con confidencialidad e integridad asegurada en la transacción).

La implementación en la parte servidora (la tienda o banco por lo general) es un poco más compleja. En primer lugar, es obligatoria la obtención de un Certificado Digital para el vendedor o para el servidor seguro, solicitándolo a una Autoridad Certificadora de prestigio reconocido, conveniendo, si es posible, que dicha autoridad sea Verisign, ya que la misma está considerada como de toda confianza por los navegadores cliente, por lo que viene activada por defecto en los navegadores cliente.

Ya con el servidor certificado, el usuario podrá realizar su compra. En el momento del pago, el vendedor obtiene el PIN de la tarjeta de crédito del cliente, la fecha de caducidad y sus datos personales (si el pago se realiza por este método), por lo que deberá disponer de algún sistema que permita el envío de estos datos a una entidad financiera capaz de realizar la transferencia bancaria necesaria para completar el pago.

Existen en España diferentes entidades bancarias y financieras que ofrecen estos sistemas a los comerciantes, realizándose la comunicación entre comerciante y banco a través de un protocolo seguro privado en la mayoría de los casos, de forma similar a lo que ocurre cuando pagamos en una tienda "real" con nuestra tarjeta de crédito. Estos sistemas se suelen conocer con el nombre genérico de **Pasarelas de Pago**.

Un sistema de pasarela más avanzado es el denominado **TPV, Terminal de Punto de Venta**. En el mismo se conecta una terminal especial al servidor web del vendedor, y mediante un software basado en script CGI se realiza la comunicación segura entre ellos.

Existen en la actualidad diferentes versiones del conjunto de protocolos SSL que se pueden implementar en los distintos servidores y que corren bajo los sistemas operativos más comunes (IIS en Windows NT-2000-XP, Apache en Unix, etc.).

VENTAJAS E INCONVENIENTES DE SSL

La tecnología basada en los protocolos Secure Socket Layer proporcionó grandes avances en la implantación de sistemas de comunicación seguros, que han hecho posible un crecimiento importante en las transacciones por Internet. Si estudiamos SSL desde el punto de vista de las bases necesarias para considerar una comunicación segura podemos sacar las siguientes conclusiones:

1. Autenticidad: SSL requiere para su funcionamiento la identificación del servidor web ante el cliente y la realiza adecuadamente, pero normalmente no se produce una identificación en sentido contrario. Es decir, no es obligada en la mayoría de los casos la presencia del certificado del usuario que se está conectando al servidor.

Por ejemplo, una de las aplicaciones más comunes de SSL es el de las aplicaciones bancarias. Cuando nos conectamos a la página web de nuestro banco para consultar las cuentas o realizar alguna operación, el servidor web tan sólo nos pide las contraseñas de acceso, lo que conlleva los típicos problemas a la hora de manejar claves: cambiarlas cada cierto tiempo, mantenerlas bien protegidas, elegir las adecuadamente, etc. Y el tema se complica cuando tenemos que seguir las mismas precauciones con cada una de las diferentes claves que los diferentes bancos y servidores seguros nos requieren.

Otro de los usos comunes de SSL es la protección de números de tarjetas de crédito o débito en compras por Internet. Pero como no se exige el uso del Certificado de Cliente, cualquier persona que obtenga el número de nuestra tarjeta y unos pocos datos personales nuestros puede realizar compras en nuestro nombre. Esto conlleva el tener que prestar mucha atención a los resguardos de nuestras

operaciones en cajeros automáticos, a desconfiar cuando un empleado de una tienda o cafetería desaparece con nuestra tarjeta para cobrar el importe de nuestra compra, etc.

Este es precisamente uno de los tipos de fraude más comunes y que causa mayores pérdidas a las compañías de crédito, lo que origina que éstas añadan una comisión en las compras bastante elevada (sobre un 5%), lo que incrementa el precio final del producto a la venta.

2. Confidencialidad: SSL proporciona una buena seguridad de que los datos no van a ser capturados por extraños de forma útil en el proceso de transferencia de los mismos, pero no proporciona ninguna seguridad después de finalizar la conexión.

Supongamos que realizamos una compra por Internet, para la cual enviamos los datos de nuestra tarjeta de crédito mediante SSL. Dichos datos quedan en poder del responsable de la tienda, que normalmente los almacena en una base de datos. Con ello, el número de nuestra tarjeta y demás datos quedan en un medio que no controlamos y que no tiene porqué ser seguro, pudiendo tener acceso a los mismos cualquier empleado de la tienda, un hacker que entre en el ordenador en el que reside la base de datos, etc.

3. Integridad: ocurre algo parecido a lo anterior. En el corto proceso que dura el envío de datos sí podemos estar seguros de que éstos no van a ser modificados, puesto que SSL lo impide. Pero una vez que finaliza la conexión segura no podemos estar tranquilos.

Imaginemos ahora que tras realizar nuestra compra el responsable de la tienda decide cambiar los datos del pedido, y en vez de enviarnos una regrabadora a 30.000 pesetas nos envía 5 a 40.000 pesetas. ¿Qué podemos hacer cuando nos llegen a casa las regrabadoras y la factura del banco?. Nada, protestar, patear y llorar, pero nada más, ya que no hay ningún recibo válido del pedido que hicimos.

4. No Repudio: en este aspecto SSL falla al máximo, ya que no hay por defecto establecido ningún método para dejar constancia de cuándo se ha realizado una operación, cuál ha sido y quiénes han intervenido en ella. SSL no proporciona formas de emitir recibos válidos que identifiquen una transacción.

Vamos ahora a suponer que realizamos un pedido a una tienda on-line, un ordenador por ejemplo, y que cuando nos llega a casa decimos que nosotros no hemos hecho ninguna compra, devolvemos el ordenador y requerimos la devolución del dinero. ¿Cómo puede demostrar el comerciante que en verdad le hicimos el pedido?. Mediante SSL, de ninguna forma.

A todo esto hay que añadir que SSL sólo proporciona seguridad en la transacción cliente-servidor seguro, pero queda otra fase de la transacción, la que va desde el servidor seguro a la empresa emisora de la tarjeta de crédito, y sobre ésta no tenemos ningún tipo de control.

Con SSL toda la seguridad de la transacción recae en la confianza que el cliente tenga en el vendedor, pues en las manos del mismo está el ser honrado y no realizar ningún fraude con los datos obtenidos y en la posterior entrega del producto comprado. Por este motivo, sólo las empresas con una honradez demostrada podrán a priori ganarse la confianza de los potenciales clientes.

Vemos pues que SSL carece de muchos de los elementos necesarios para construir un sistema de transacciones seguras usando Internet. Para intentar paliar estos fallos se han intentado sacar al mercado y estandarizar otros sistemas diferentes, como SET, que veremos a continuación, pero el caso es que hasta ahora ninguno de ellos ha conseguido desplazar a SSL. ¿Porqué?.

Tal vez sea porque, a pesar de sus fallos, SSL es una tecnología rápida, fácil de implementar, barata y cómoda para el usuario, que no tiene que conocer cómo funciona, tan sólo usarla. Y desde el punto de vista del comerciante o de la empresa que le facilita el hosting, SSL es igualmente sencillo de implementar, no precisando de servidores de especiales características.

OTROS PROTOCOLOS SEGUROS

Protocolo TLS - Transport Layer Security.-

Para intentar corregir las deficiencias observadas en SSL v3 se buscó un nuevo protocolo que permitiera transacciones seguras por Internet, sobre todo teniendo en cuenta que SSL es propiedad de la empresa Netscape. El resultado de esta búsqueda fue el protocolo TLS, que permite una compatibilidad total con SSL siendo un protocolo público, estandarizado por el IETF.

TLS busca integrar en un esquema tipo SSL al sistema operativo, a nivel de la capa TCP/IP, para que el efecto "túnel" que se implementó con SSL sea realmente transparente a las aplicaciones que se están ejecutando. Parte de las mismas bases que SSL, pero se diferencia de él en varios aspectos fundamentales:

1. En el paso CertificateRequest del protocolo Handshake los clientes sólo contestan con un mensaje si son SSL.
2. Las claves de sesión se calculan de forma diferente.
3. A la hora de intercambiar las claves, TLS no soporta el algoritmo simétrico Fortezza, que sí es soportado por SSL. Esto es debido a la búsqueda de un código público, ya que Fortezza es de propiedad privada.
4. TLS utiliza dos campos más en el MAC que SSL, lo que lo hace más seguro.

A pesar de mejorar SSL y de ser público, TLS no está teniendo la aceptación que se esperaba (por lo menos por ahora).

Protocolo S-HTTP.-

El protocolo Secure HTTP fue desarrollado por Enterprise Integration Technologies, EIT, y al igual que SSL permite tanto el cifrado de documentos como la autenticación mediante firma y certificados digitales, pero se diferencia de SSL en que se implementa a nivel de aplicación. Se puede identificar rápidamente a una página web servida con este protocolo porque la extensión de la misma pasa a ser .shtml en vez de .html como las páginas normales.

El mecanismo de conexión mediante S-HTTP, que ahora se encuentra en su versión 1.1, comprende una serie de pasos parecidos a los usados en SSL, en los que cliente y servidor se intercambian una serie de datos formateados que incluyen los algoritmos criptográficos, longitudes de clave y algoritmos de compresión a usar durante la comunicación segura.

En cuanto a estos algoritmos, los usados normalmente son RSA para intercambio de claves simétricas, MD2, MD5 o NIST-SHS como funciones hash de resumen, DES, IDEA, RC4 o CDMF como algoritmos simétricos y PEM o PKCS-7 como algoritmos de encapsulamiento.

A diferencia de SSL, el protocolo S-HTTP está integrado con HTTP, actuando a nivel de aplicación, como ya hemos dicho, negociándose los servicios de seguridad a través de cabeceras y atributos de página, por lo que los servicios S-HTTP están sólo disponibles para el protocolo HTTP. Recordemos que SSL puede ser usado por otros protocolos diferentes de HTTP, pues se integra a nivel de socket.

8 – PROTOCOLO SET

Ya hemos visto cómo SSL adolece de graves defectos a la hora de implementar las cuatro condiciones básicas de una transacción segura. Estas carencias hicieron que diferentes empresas y organismos buscaran un nuevo sistema que permitiera realizar operaciones sensibles por Internet de forma segura, con el objeto de estimular la confianza de los consumidores en el comercio electrónico.

En febrero de 1996 un grupo de empresas del sector financiero, informático y de seguridad (Visa International, MasterCard, Microsoft, Netscape, IBM, RSA, ect.) anunciaron el desarrollo de una nueva tecnología común destinada a proteger la compras a través de redes abiertas como Internet basadas en el uso de tarjetas de crédito. Esta nueva tecnología se conoce con el nombre de Secure Electronic Transactions (Transacciones Electrónicas Seguras), SET, y ha sido creada exclusivamente para la realización de comercio electrónico usando tarjetas de crédito.

SET se basa en el uso de certificados digitales para asegurar la perfecta identificación de todas aquellas partes que intervienen en una transacción on-line basada en el uso de tarjetas de pago, y en el uso de sistemas criptográficos de clave pública para proteger el envío de los datos sensibles en su viaje entre los diferentes servidores que participan en el proceso. Con ello se persigue mantener el carácter estrictamente confidencial de los datos, garantizar la integridad de los mismos y autenticar la legitimidad de las entidades o personas que participan en la transacción, creando así un protocolo estandar abierto para la industria que sirva de base a la expansión del comercio electrónico por Internet.

Las especificaciones formales del protocolo SET 1.0 se hicieron públicas el 31 de mayo de 1997, y se pueden encontrar en el sitio web oficial de SETco, <http://www.setco.org>, organismo encargado de homologar los módulos de programación y los certificados desarrollados por empresas privadas que se usen en implementaciones del protocolo SET.

Como características principales de SET podemos destacar:

- Es un estándar abierto y multiplataforma, en el que se especifican protocolos, formatos de mensaje, certificados, etc., sin limitación alguna respecto al lenguaje de programación, sistema operativo o tipo de máquina usados.
- Su principal objetivo es la transferencia segura de números de tarjetas de crédito.
- Utiliza codificación estándar (ASN.1 y DER).
- Es independiente del medio de comunicación utilizado. Fue diseñado para su uso en Internet, pero permite la conexión a través de cualquier tipo de red siempre que se definan los interfaces adecuados. Además, el protocolo SET se puede transportar directamente mediante TCP, mediante correo electrónico basado en SMTP o MIME y mediante HTTP en páginas web.
- Utiliza estándares criptográficos reconocidos y ampliamente usados (PKCS, Certificados X.509, etc.).
- El formato de los mensajes usados está basado en el estándar PKCS-7, al igual que SSL y S-MIME.
- Se basa en el uso de la Criptografía de Clave Pública.
- Realiza una Autenticación de todas las partes participantes en la transacción usando certificados digitales.

Vimos que en el proceso SSL sólo intervienen dos entidades: el Comprador (Cardholder) y el Vendedor (Merchant). Pues bien, SET incluye otras entidades adicionales necesarias para la transacción:

- La **Pasarela de Pago** (Gateway Payment), que permite la comunicación directa a través de Internet entre el comerciante y las Redes Bancarias, con lo que el papel del vendedor queda limitado

a un mero intermediario entre el cliente y su banco. Puede ser una entidad independiente o el mismo banco del comerciante.

- El Banco o entidad financiera (Issuer) que ha emitido la tarjeta de crédito que va a usar el cliente en el proceso de pago.

- El Banco del comerciante (Acquirer), en el que éste tiene su cuenta.

Además de estas entidades principales existen otras dos relacionadas con ellas:

- La empresa propietaria de la marca de la tarjeta de crédito, como Visa, MasterCard, American Express, etc., que avalan las tarjetas.

- **Autoridades de certificación**, que emiten los certificados digitales usados como medio de autenticación de las entidades que intervienen directamente en la operación. Pueden ser entidades independientes autorizadas, bancos o los mismos propietarios de la marca de la tarjeta. El papel de éstas y sus diferentes modalidades lo veremos más adelante.

Proceso de pago con SET.-

El proceso de pago en una transacción electrónica usando el protocolo SET admite un gran número de opciones diferentes pero, básicamente, consta de los siguientes pasos:

1. El cliente, tras seleccionar los artículos a comprar en el sitio web del vendedor, envía a éste un formulario de pedido, siendo respondido por el comerciante con el envío de su certificado digital y el de la pasarela de pago. El cliente comprueba la validez de los certificados y envía entonces al comerciante una **orden de pago**, que está dividida en dos secciones o documentos diferentes: la **Información de pedido** (OI), en la que figuran los datos de los productos comprados, su precio y las demás informaciones necesarias para la compra, y la **Instrucción de compra** (PI), en donde se describen sus datos bancarios y se dan instrucciones para el pago a la entidad vendedora.

2. Esta orden de pago se firma digitalmente por medio de un algoritmo especial, denominado **Firma Dual**, que se realiza concatenando primero los resúmenes hash de los dos documentos generados y encriptando esta concatenación después con su llave privada, para seguidamente encriptar la Firma Dual mediante una clave simétrica generada por su software SET. Por último, se encriptan la clave simétrica generada y el número de la tarjeta de crédito con la llave pública de la pasarela de pago.

De esta forma el vendedor no puede conocer los datos bancarios del comprador, y el banco no puede conocer la información sobre los productos comprados, a pesar de que ambos documentos están ligados por la misma firma. En ciertos casos es posible realizar la transacción sin esta firma dual, estableciéndose mediante un protocolo inicial qué método se va a usar.

3. El vendedor recibe la orden de compra y la firma dual del cliente, se queda con la descripción de la compra y tras comprobar la autenticidad del comprador, utilizando para ello la firma digital de éste y su certificado, y la integridad de los datos recibidos envía los datos financieros a la Pasarela de Pago encriptados con la clave pública de la misma.

4. La Pasarela de Pago comprueba la autenticidad del comprador y la integridad del PI del mismo, y con el mensaje del vendedor comprueba la relación existente entre la descripción de la compra enviada al vendedor y la usada para la firma dual recibida.

5. Si todo es correcto, la Pasarela de Pago envía mediante las redes de comunicación bancarias el PI al banco del vendedor y solicita autorización para realizar el pago, mediante un documento denominado **Petición de autorización de pago**.

6. El banco del vendedor comprueba entonces que la tarjeta de crédito es válida y permite el cargo del importe de la compra, enviando entonces un documento a la pasarela, denominado **Autorización de pago**, que autoriza el proceso de compra.

7. Una vez informado el vendedor de la autorización procede al envío de los artículos comprados al cliente, y después de la entrega física del producto pide el importe de la venta a la Pasarela de Pagos, proceso que se conoce con el nombre de **Solicitud de pago**.

8. Entonces la Pasarela de Pagos pide al banco del comprador la transferencia del importe de la venta al banco del vendedor, petición que recibe el nombre de **Solicitud de compensación**. Entonces se le hace efectivo al vendedor el importe, con lo que se cierra el proceso total de compra.

Todos los documentos implicados en el proceso anterior deben llevar un número identificador único de transacción, conocido como **ID**.

* El proceso completo de pago mediante el protocolo SET podéis observarlo en el siguiente gráfico:



9 - SET – MONEDEROS ELECTRONICOS

Con objeto de facilitar los pagos en transacciones basadas en el protocolo SET se desarrolló una aplicación especial, denominada monedero electrónico, que simula la funcionalidad de una cartera tradicional, y que permite al usuario cliente disponer de un lugar en el que guardar los números de sus tarjetas de crédito y los resguardos de las compras realizadas.

Estos monederos o carteras, conocidos también con el nombre de **Wallets**, se pueden integrar en la actualidad en cualquiera de los navegadores estándar, y permiten a los usuarios realizar compras por Internet de forma cómoda y segura, usando el protocolo SET. Los datos de las tarjetas de crédito y de la compra se transmiten y se almacenan de forma segura (encriptados con un sistema simétrico), garantizando la Autenticidad y la Confidencialidad en el proceso de compra de forma totalmente transparente al usuario, que sólo se debe preocupar de elegir los productos que deséa adquirir y decir con qué tarjeta de las contenidas en el monedero deséa pagarlos.

Los monederos electrónicos poseen un sistema de administración propio que permite al usuario la cómoda gestión de sus tarjetas y de los resguardos de las compras que ha realizado. Además, son programas de poco peso, lo que permite al usuario guardarlos en un disquete y llevarlo consigo para poder operar con ellos en cualquier ordenador, sin que quede constancia luego en el mismo de ninguna de las transacciones realizadas (todo se guarda en el monedero, nada queda en fuera de él). El acceso a los datos de monedero se encuentra protegido con una contraseña propia de cada usuario, lo que permite que una única cartera pueda tener varios usuarios distintos, cada uno con su clave de acceso, sus propias tarjetas y sus propios resguardos de compra, no pudiendo en ningún momento un usuario acceder a los datos de otro.

La misiones principales del monedero electrónico son comunicarse automáticamente con la aplicación de venta del comerciante, guardar la información sobre las tarjetas de crédito y las compras realizadas y manejar los certificados usados en la transacción para autenticar a las partes que intervienen en la misma. También se encargan de la recuperación de transacciones perdidas, de forma que si en el proceso de compra se interrumpe la conexión a Internet, posteriormente puede continuar ésta en el punto en que se quedó.

Como utilidad adicional, los monederos permiten la personalización de su interfaz con logotipos y textos propios de cada usuario u organización que deséa distribuirlos entre sus clientes.

SET – TERMINALES PUNTO DE VENTA VIRTUALES

En el comercio tradicional con tarjetas de crédito existe un mecanismo (una aplicación), denominada Terminal Punto de Venta, cuya misión es solicitar la autorización de pago mediante la tarjeta de crédito del cliente. Todos conocemos este sistema, consistente en una pequeña máquina, comunicada con la pasarela de pago por vía telefónica, en la que el vendedor pasa la banda magnética de nuestra tarjeta y recibe la autorización para la venta tras comprobarse la validez de la tarjeta y la disponibilidad de fondos asociados a la misma.

SET proporciona una aplicación parecida, denominada Terminal Punto de Venta Virtual, que funciona de forma transparente al usuario. Cuando éste ordena una compra, generalmente usando su monedero electrónico, en el proceso de pago que se origina el Terminal Punto de Venta Virtual almacena la información de la transacción y establece contacto con el sistema financiero usado, a través de la Pasarela de Pago usada por SET, que será la encargada de autorizar la compra.

Además, el TPV Virtual dispone de una aplicación de administración que permite llevar un control total sobre todas las transacciones que el sistema haya procesado, y mediante la cual es posible:

- Comprobar en todo momento los certificados digitales de que dispone el sistema, así como su gestión.
- Configurar las conexiones con los otros sistemas.
- Controlar en todo momento el estado de las compras.

- Almacenar ficheros logs con los eventos que se producen en el sistema, mediante los cuales el administrador puede localizar rápidamente cualquier anomalía.
- Gestionar batches, que permiten al comerciante conocer en todo momento las compras ya liquidadas y comprobar si los datos almacenados en su sistema coinciden con los correspondientes de la entidad financiera.
- Generar informes estadísticos basados en diferentes criterios, que permiten tener un control gráfico sobre todos los datos.

SET – PASARELAS DE PAGO

Las Pasarelas de Pago en SET son las encargadas de conectar a los comerciantes con las entidades financieras. Reciben peticiones de autorización, liquidación o reconciliación de pagos de los sistemas comerciales TPV Virtuales y las encaminan hacia los sistemas autorizadores de pago tradicionales.

El encaminamiento de peticiones financieras provenientes de SET hacia el sistema autorizador se realiza a través de un módulo dinámico, de forma que resulta posible la conexión de la aplicación con cualquier sistema autorizador existente en el mercado, independientemente del formato o protocolo de comunicaciones usado.

VENTAJAS E INCONVENIENTES DE SET

No cabe duda que el sistema SET proporciona buenas cualidades de seguridad, integridad, autenticidad y no rechazo en las transacciones comerciales por redes abiertas basadas en el pago mediante tarjetas de crédito, y que existe un gran empuje por parte de las principales empresas financieras y expendedoras de tarjetas de crédito para estandarizar su uso, pero el caso es que no se ha logrado un desarrollo e implementación masivo del mismo.

Uno de los factores que tal vez hayan influido más en ésta pasividad del mercado respecto a SET es el de la complejidad intrínseca del mismo. Con SSL el usuario no tiene que hacerse con certificado alguno (normalmente), ni tiene que andar instalando en su ordenador software adicional; tan sólo debe seleccionar los productos que desea comprar y aceptar el pago.

Otro factor a tener en cuenta es la relativa lentitud de proceso de SET, al tener que realizarse diferentes verificaciones de identidad e integridad por parte de diversas entidades a lo largo de una transacción.

Por todo esto, y aunque cada vez se pueden encontrar más tiendas virtuales que usan SET, sigue siendo SSL el protocolo más usado en las transacciones por Internet.

10 - SET - CERTIFICACIONES

El protocolo SET es constituye el primer proyecto de certificación a escala global que se va a realizar. Los certificados SET se estructuran siguiendo una jerarquía piramidal única, cuya cúspide la ocupa la **Autoridad Certificadora Raiz** (Root CA), que es la encargada de certificar a todas las demás autoridades certificadoras.

Bajo la Autoridad Certificadora Raiz se encuentran las Brand CA, o **Autoridades de Certificación de Marca**, propiedad de las entidades emisoras de tarjetas de crédito, entre las que destacan las pertenecientes a Visa International y MasterCard International, propulsoras del protocolo SET. Estas entidades son certificadas por la Autoridad Certificadora Raiz. Por lo tanto, cuando una Brand CA desea obtener un certificado SET debe realizar una petición a la Root CA, en un archivo de formato estándar, el PKCS#107. Si dicha solicitud es aprobada se genera otro archivo de respuesta, denominado PKCS#7, que es remitido a la CA solicitante.

Las Autoridades de Certificación de marca se encargan principalmente de emitir certificados SET a Autoridades de Certificación Geopolítica. También son responsables de la generación de los archivos BCI de certificados revocados, recopilando para ello las CRL de las CA por debajo de ellas. Una vez confeccionados estos archivos son enviados a las Geopolitical CA, que son las encargadas de su distribución.

Las Brand CA pueden autorizar a su vez a otras entidades, denominadas Geopolitical CA o **Autoridades de Certificación Final**, para que funcionen como autoridades certificadoras. Un ejemplo de entidad de este tipo es **ACE, Agencia de Certificación Española**. Las Autoridades de Certificación Final se encargan de emitir los certificados SET a los usuarios finales del sistema, clientes, vendedores y pasarelas de pago.

A la hora de obtener un certificado SET se requiere un proceso de autenticación de los datos que en él van a figurar, al igual que sucede con los certificados X.509 v3. En el caso de SET la verificación de datos corresponde a unas entidades creadas al efecto, que se denominan **Autoridades de Registro**.

Su labor es la actuar como avaladores ante la CA de los usuarios que solicitan el certificado, encargándose también de tramitar los mismos. Las entidades destinadas a asumir el papel de Autoridades de Registro son los propios bancos, permitiendo con ello que los certificados estén asociados a cuentas bancarias y no a personas físicas, con lo que hace posible la compra anónima por Internet, al no aparecer en ningún momento el nombre del cliente que va a efectuar el pedido.

Las Autoridades de Registro actúan en nombre y por cuenta de la Autoridad de Certificación correspondiente, y deben superar un proceso de homologación antes para garantizar su fiabilidad. Sus principales misiones son validar solicitudes de certificado en base a determinados procedimientos de identificación según el tipo de certificado, solicitar luego el correspondiente certificado a la Autoridad Certificadora y entregar el mismo, una vez obtenido, al usuario final del mismo, usando para ello un disquete u otro soporte adecuado.

Toda Autoridad de Registro debe tener a disposición de los solicitantes un documento, denominado **Prácticas de Registro**, que especifique claramente los procedimientos operativos y de garantía de seguridad que exige y facilita.

Este sistema jerárquico y modular de las Autoridades de Certificación proporciona una gran flexibilidad a SET para adaptarse a todo tipo de necesidades, desde la pequeña empresa que tan sólo desea certificar a sus compradores hasta la gran empresa que quiere ofrecer a sus clientes una solución SET completa.

LA AGENCIA DE CERTIFICACIÓN ESPAÑOLA

La Agencia de Certificación Española se constituyó en 1997, y está formada por el Grupo Telefónica (40%), SERMAPA-Visa España (20%), CECA (20%) y Sistema 4B (20%). Su misión principal es dar respuesta a las necesidades derivadas de la implementación de protocolos seguros para transacciones por Internet por medio de la emisión de certificados electrónicos, documentos capaces de facilitar el intercambio de información en redes telemáticas abiertas bajo las máximas garantías de seguridad para todas las partes, proporcionando también una infraestructura tecnológica sustentada en la tecnología de clave pública (PKI).

Pretende con ello constituirse en una parte mediadora de confianza entre usuarios, vendedores y entidades financieras. Los certificados SET de ACE se emiten al titular de la tarjeta de crédito o débito, al comercio, a la pasarela de pago y a las entidades financieras emisora y adquirente, garantizando con ellos la total seguridad e intimidad en el proceso de compra.

Tipos de certificados ACE.-

Al igual que ocurre con los certificados X.509 v3, los certificados SET expedidos por ACE pueden ser de diferentes tipos, en función del grado de confianza exigido para cada tipo de operación. Buscando una claridad formal en los certificados y en sus posibles usos, éstos se han dividido en categorías y subcategorías, de las que las más importantes son:

1. Certificados de Categoría 0 o Intranet: son emitidos bajo responsabilidad exclusiva de una entidad u organización, que actúa como Autoridad Certificadora, para usuarios dentro de su ámbito de acción, nunca para su uso por terceros que deban confiar en la existencia de tales usuarios. Se exige que la entidad emisora de este tipo de certificados haya recibido previamente un certificado de Categoría 1A. Estos certificados presentan dos modalidades: entidad no comprobada y entidad comprobada mediante información en base de datos, y en ambos casos es total y única responsabilidad de la entidad emisora la veracidad de la información contenida en los certificados.

2. Certificados de Categoría 1 o Privada: que a su vez se dividen en siete tipos diferentes:

- 1A Particulares: que aseguran la identidad de los suscriptores particulares mediante identificación de los mismos ante una Autoridad Certificadora o ante una Autoridad de Registro homologada, presentando documentación suficiente para demostrar su identidad personal.

- 1B Profesionales: permiten asegurar la identidad de suscriptores que ejercen como profesionales independientes (propietarios de empresas individuales o personas con poderes otorgados por dicho profesional), siendo necesario presentar ante la Autoridad Certificadora o ante la Autoridad de Registro homologada documentación suficiente que demuestre su condición de profesionales, además de los requisitos necesarios para el certificado de clase 1A.

- 1Ca Personal de empresas: que aseguran la existencia y denominación de sujetos que trabajan en diversas entidades de derecho público y privado (empresas, corporaciones y agencias gubernamentales), con especificación del departamento al que pertenecen y del cargo que ocupan. Se requiere para su expedición documentación que asegure su posición dentro de la empresa, acreditada por el representante legal de la entidad ante la Autoridad Certificadora o ante la Autoridad de Registro homologada.

- 1Cb Departamentos de empresa: aseguran la existencia y denominación de departamentos de entidades de derecho públicas y privadas, con las mismas garantías de certificación que en el caso anterior.

- 1C c Master técnico: que aseguran la responsabilidad del gestor, responsable o administrador de sistemas de una entidad concreta, siendo el responsable de los certificados emitidos para dicha empresa y la persona ante la que hay que acudir en referencia a las cuestiones técnicas que se planteen en los sistemas web de comercio electrónico.

- 1D Entidad: aseguran la existencia y denominación de diversas entidades de derecho públicas y privadas de forma análoga a los certificados anteriores, pero con mayor grado de garantías, ya que el

titular del certificado de Entidad es una persona que ostenta poderes dentro de la misma, pudiendo obligar a dicha Entidad de acuerdo con esos poderes, contenidos en los estatutos de la misma.

- 1S Servidor seguro: emitidos únicamente a empresas, aseguran la existencia y denominación de empresas, agencias gubernamentales y corporaciones en Internet y otras redes. Implican la identificación de un servidor seguro, asociándolo con la empresa responsable del mismo y con la URL (dirección de Internet) por la que se debe acceder al servidor, elementos que debe acreditar su representante legal ante la Autoridad Certificadora o ante la Autoridad de Registro homologada.

Los certificados de Categoría 1A, 1B, 1C en todas sus variantes y 1D son utilizados por los suscriptores para garantizar frente a terceros su identidad, la autenticidad e integridad de sus mensajes y para cifrar y firmar mensajes, empleándose también para permitir la identificación del suscriptor en aplicaciones de servidor seguro HTTP con SSL v3 activado.

Los certificados de categoría 1S se usan para aplicaciones de banca electrónica, comercio electrónico y en todas otras aquellas aplicaciones que precisan la identificación del servidor seguro que interviene en las mismas, tanto en entornos de Internet como en Intranet.

Los certificados de categoría 1 tienen naturaleza de documento privado, no pudiéndose usar en aplicaciones como la firma de código binario.

3 . Certificados de Categoría 2 o Categoría Oficial, que se diferencian de los de la Categoría 1 en que el proceso de identificación debe realizarse ante una entidad u organismo oficial, y que se presentan seis tipos diferentes:

- 2B Empresa individual: permiten asegurar mediante la intervención de un organismo oficial (como una Cámara de Comercio) la identidad de los suscriptores que ejercen como empresas individuales, exigiendo para la identificación que éstos presenten la documentación suficiente para ello.

- 2Ca Simple: aseguran la existencia y denominación de sujetos que trabajan en diversas entidades de derecho público y privado (empresas, corporaciones, etc.), con especificación del departamento y cargo que ocupan, elementos que debe acreditar el representante legal de la entidad, previamente, ante un organismo oficial.

- 2Cb Departamentos de empresas: aseguran la existencia y denominación de los departamentos de las diversas entidades de derecho público y privado citadas anteriormente, con las mismas garantías que en el caso de certificación simple, pero realizada ante un organismo oficial.

- 2Cc Master técnico: que aseguran la responsabilidad del gestor, responsable o administrador de sistemas de una entidad concreta, siendo el responsable de los certificados emitidos para dicha empresa y la persona ante la que hay que acudir en referencia a las cuestiones técnicas que se planteen en los sistemas web de comercio electrónico.

- 2D Sociedad: aseguran la existencia y denominación de diversas entidades de derecho públicas y privadas de forma análoga a los certificados anteriores, con mayor grado de garantías que en el caso de certificados de categorías inferiores, ya que el titular del Certificado de Sociedad es la persona que ostenta poderes dentro de la misma, pudiendo obligar a dicha Entidad de acuerdo con esos poderes, contenidos en los estatutos de la sociedad.

- 2S Servidor seguro: emitidos únicamente a empresas, aseguran la existencia y denominación de empresas, agencias gubernamentales y corporaciones en Internet y otras redes. Implican la identificación de un servidor seguro, asociándolo con la empresa responsable del mismo y con la URL (dirección de Internet) por la que se debe acceder al servidor, elementos que debe acreditar su representante legal ante un organismo oficial competente.

Los usos de estos certificados son análogos a los de la Categoría 1, pero son más estrictos a la hora de verificar la veracidad de los datos que en ellos figuran.

4. Certificados de Categoría 3, que ofrecen los mayores niveles de comprobación y seguridad, ya que son validados por un fedatario público, y de los que existen seis modalidades:

- 3A Particulares: que ofrecen las medidas más importantes identidad de los suscriptores particulares mediante identificación de los mismos ante un fedatario público, presentando documentación suficiente para demostrar su identidad personal.

- 3B Profesionales: permiten asegurar la identidad de suscriptores que ejercen como profesionales independientes (propietarios de empresas individuales o personas con poderes otorgados por dicho profesional), siendo necesario presentar ante un fedatario público documentación suficiente que demuestre su condición de profesionales, además de los restantes datos de identidad personal.

- 3Ca Personal de empresa: que aseguran la existencia y denominación de sujetos que trabajan en diversas entidades de derecho público y privado (empresas, corporaciones y agencias gubernamentales), con especificación del departamento al que pertenecen y del cargo que ocupan. Se requiere para su expedición documentación que asegure su posición dentro de la empresa, acreditada por el representante legal de la entidad, previamente, ante un fedatario público.

- 3Cb Departamentos de empresas: aseguran la existencia y denominación de los departamentos de las diversas entidades de derecho público y privado citadas anteriormente, con las mismas garantías que en el caso de certificación simple, pero realizada ante un fedatario público.

- 3Cc Master técnico: que aseguran la existencia de un gestor, responsable o administrador de sistemas de una entidad concreta, siendo el responsable de los certificados emitidos para dicha empresa y la persona ante la que hay que acudir en referencia a las cuestiones técnicas que se plantéen en los sistemas web de comercio electrónico. La documentación de identificación necesaria se debe presentar previamente ante un fedatario público.

- 3D Entidad: aseguran la existencia y denominación de diversas entidades de derecho públicas y privadas de forma análoga a los certificados anteriores, pero con mayor grado de garantías, ya que el titular del certificado de Entidad es una persona que ostenta poderes dentro de la misma, pudiendo obligar a dicha Entidad de acuerdo con esos poderes, contenidos en los estatutos de la misma.

Los certificados de Categoría 3 son empleados por los suscriptores cuando la Ley exija la intervención de un fedatario público en la transacción, y suponen el mayor nivel posible de autenticación previa del suscriptor del certificado. En este caso, los certificados gozan de la naturaleza jurídica de documentos públicos. Los documentos firmados son, en todo caso, documentos privados.

11 – SERVIDORES SEGUROS

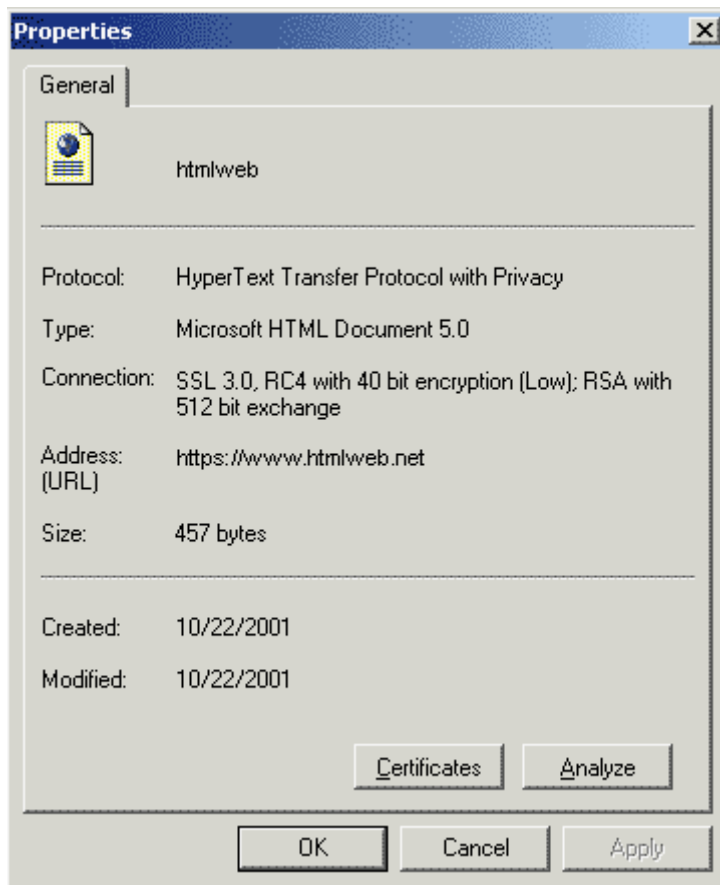
Se entiende por Servidor Seguro un servidor de páginas web que establece una conexión cifrada con el cliente que ha solicitado la conexión, de manera que nadie, salvo el servidor y el cliente, puedan tener acceso a la información transmitida de forma útil.

El uso de servidores seguros es un elemento imprescindible en todos aquellos servicios que utilicen información confidencial, como operaciones bancarias on-line, compras por Internet, acceso a servidores de datos sensibles, etc.

Para conseguir la confidencialidad e integridad de datos perseguida los servidores seguros se basan en el uso de sistemas criptográficos mixtos, que combinan la Criptografía de clave pública con la de clave simétrica. Pero esta protección que debiera darnos la Criptografía es, en la práctica, difícil de encontrar, debido a las severas leyes de exportación de software de cifrado que impone el gobierno de EEUU, sobre todo en lo que respecta a la longitud de las claves que usan. Para garantizar al usuario su autenticidad, los servidores seguros hacen uso de los certificados digitales ya estudiados.

Cuando accedemos a un servidor seguro normalmente nos aparece una ventana indicándonos que vamos a iniciar una conexión segura, y el candado situado en la parte inferior de la ventana del navegador aparecerá cerrado cuando entremos a la página segura (Atención: la presencia del candado cerrado no garantiza una comunicación segura; hace falta comprobar el certificado del servidor). Además, si miramos en la barra de direcciones veremos que ahora estamos usando el protocolo HTTPS, que corresponde al protocolo HTTP con privacidad.

Podemos acudir al menú "Archivo" > "Propiedades" para obtener información más detallada sobre el documento seguro, entre la que destaca:



- Protocolo: HTTP Seguro (con privacidad).

- Conexiones: Protocolo Secure Socket Layer versión 3.0, algoritmo simétrico de cifrado RC4 con longitud de claves de 40 bits, algoritmo de cifrado de clave pública RSA de 512 bits de longitud de clave.

- URL del servidore seguro.

Como vemos, tenemos una longitud de clave RC4 de 40 bits, limitación impuesta por el gobierno de EEUU al sistema de cifrado para su exportación. Esta longitud de clave no es todo lo segura que debiera para ser usada en transacciones delicadas; ya ha sido violentada anteriormente, y aunque ésto no significa que pueda serlo en el tiempo de duración de la conexión segura, si es un indicio de la debilidad del sistema con esas longitudes de clave.

También podemos acceder desde una página segura al certificado digital del servidor de forma rápida. Para ello basta hacer seleccionar el botón "Certificates" de la ventana anterior o hacer doble click sobre el candado cerrado.

Otro aspecto importante a considerar son los fallos a la hora de implementar los protocolos criptográficos, sobre todo en lo que respecta a la configuración propia del servidor web seguro y a los fallos de implementación que de los protocolos hacen los navegadores cliente. Uno de estos fallos es la relativa falta de seguridad de los números pseudoaleatorios generados para el proceso de creación de claves durante la fase Handshake.

A pesar de todas estas consideraciones no hay que ser alarmista en cuanto al uso de los servidores y protocolos seguros, ya que generalmente el tiempo de duración de la conexión segura es lo suficientemente pequeño como para resultar imposible, con los medios actuales, descifrar las claves en un tiempo útil. Y a esto hay que añadir las innumerables ventajas que obtenemos de este tipo de aplicaciones, que permiten realizar transacciones seguras por Internet.

Para minimizar los riesgos posibles, a la hora de implementar o aceptar un servicio de servidor seguro podemos exigir que se cumplan una serie de condiciones, entre las que podemos destacar:

* Que el certificado de servidor seguro se corresponda con los de máximas garantías de verificación y que haya sido expedido por una Autoridad Certificadora de toda confianza. Generalmente los JAVAHISPANOnavegadores cliente reconocen como tales a VeriSign/RSA Secure Server Certification Authority, EuroSign y Thawte Server.

* Que el navegador usado en la comunicación tenga implementada la última versión de SSL, es decir, el protocolo SSL 3.0. Versiones anteriores son válidas, pero no recomendadas.

* El uso de un sistema de cifrado simétrico robusto (RC4 RC5 o similar) con longitudes de clave largas (de entre 64 y 128 bits). A pesar de la limitación clásica del gobierno USA para la exportación de sistemas con claves largas, actualmente las principales compañías fabricantes de navegadores tienen permiso para exportar las actualizaciones a cifrado de 128 bits, por lo que es conveniente la actualización de nuestros navegadores. estas actualizaciones a veces son también suministradas por bancos y entidades financieras a sus

clientes.

12 – COMO INSTALAR APACHE+SSL(+TOMCAT) EN LINUX)

12.1 – PREPARACIÓN

Suponemos que hemos bajado de internet todos los ficheros que necesitamos (servidor apache, mod_ssl, openssl, y si se quiere los de Tomcat) Suponemos que los guardamos todos en un directorio temporal, por ejemplo /tmp/www/, y tomamos ese directorio como centro de operaciones

```
# cd /tmp/www
```

12.2 – CONFIGURANDO EL SERVIDOR WEB APACHE

Lo primero que haremos, y puesto que otros pasos así lo requieren, será descomprimir nuestro servidor apache y configurarlo indicando en que directorio lo queremos instalar:

```
# tar xvzf apache_1.3.12.tar.gz
# cd apache_1.3.12
# ./config --prefix=/usr/local/apache
# cd ..
```

12.3 – INTALANDO OPENSLL

Una vez hemos descargado los ficheros de OpenSSL para Linux, lo descomprimos como siempre:

```
# tar xvzf openssl-0.9.5a.tar.gz
# cd openssl-0.9.5a
```

Lo configuramos indicando donde lo queremos instalar (opcion prefix del script config), lo compilamos y lo instalamos, como cualquier programa Linux que viene en forma de código fuente, nada nuevo:

```
# ./config --prefix=/usr/local/ssl
# make
# make test
# make install
# cd ..
```

12.4 – CONFIGURANDO MOD_SSL

Tenemos que tener cuidado al descargar el fichero que contiene los fuentes de mod_ssl y hacerlo del que corresponda con nuestra versión de Apache. Estos ficheros (por ejemplo mod_ssl-2.6.4-1.3.12.tar.gz) traen dos números de la serie. El primero (2.6.4) indica la versión de mod_ssl, el segundo indica la versión de Apache a la que corresponde (1.3.12).

Rápidamente descomprimos el mod_ssl que hemos descargado de internet, y lo configuramos indicándole donde tenemos el código fuente del servidor Apache (que aún no hemos instalado). Por ejemplo:

```
# tar xvzf mod_ssl-2.6.4-1.3.12.tar.gz
# cd mod_ssl-2.6.4-1.3.12
# ./configure --with-apache=../apache_1.3.12
# cd ..
```

12.5 – INTALANDO EL SERVIDOR WEB APACHE

En este punto es donde podremos añadir a nuestro servidor Apache todos los modulos que queramos (Perl, PHP, Tomcat, etc), de la misma forma que se indica en sus respectivas guías de instalación.

Pero bueno, en esta guía tratamos el modulo SSL y en eso estamos. Primero le indicamos donde hemos descomprimido OpenSSL y despues le indicamos los distintos modulos que queremos usar:

```
SSL_BASE=../openssl-0.9.5a ./configure --prefix=/usr/local/apache \
--enable-module=ssl --enable-shared=ssl
```

(aqui meteriamos el modulo de Tomcat (--enable-module=so), php, etc.)

Ya tenemos Apache preparado para instalarlo, así que lo compilamos:

```
# make
```

Creamos nuestro certificado y llave, contestando un montón de preguntas sobre nuestra "empresa" (recordar que tiene que tener el mismo nombre que nuestro servidor):

```
# make certificate TYPE=custom
```

Y finalmente instalamos el Apache como siempre:

```
# make install
```

Si todo ha ido bien, tendremos al final de la instalación un cuadro indicándonoslo, y mostrando lo que tenemos que escribir para ejecutar el servidor apache "normal" y "seguro".

(servidor normal)

```
# /usr/local/apache/bin/apachectl start
```

(servidor seguro)

```
# /usr/local/apache/bin/apachectl startssl
```

Si arrancamos la versión segura nos pedirá el password que hallamos dado a nuestra llave SSL, y listo, ya podemos acceder a nuestro servidor seguro desde nuestro browser con la dirección:

```
https://127.0.0.1/
```

12.6 – COMPROBAMOS QUE FUNCIONA

Simplemente dirigimos nuestro navegador, en mi caso mi flamante open source Mozilla 0.9 (descárgalo desde <http://www.mozilla.org>) a la dirección segura de nuestro servidor, como hemos dicho con:

```
https://127.0.0.1/
```

Y recibiremos la siguiente ventana informándonos de la entrada en la "zona segura" y de que el certificado no está aprobado por ningún organismo oficial, por lo que podría no ser válido.

NOTA: posiblemente, si no eres un raro como yo, y vives en España, el texto te aparecerá en otro idioma, seguramente en español, pero me temo que mi Linux habla inglés ;-). Si alguien tiene ganas, que me mande la versión en castellano de estas dos ventanas.

Aceptamos el certificado (lo examinamos si queremos), y nos sale la página de presentación de Apache con la información de SSL.

NOTA: la misma página sale aunque hallas arrancado el servidor normal, así que no te asustes. Tienes que fijarte en la dirección indicada para llegar a ella. Perdonar por la baja calidad, pero una imagen de 1024x768 era demasiado grande ;-).

13 – COMO INSTALAR APACHE+SSL(+TOMCAT) EN WINDOWS

13.1 – PALABRAS PREVIAS

Quede dicho de antemano, que hablar de un servidor seguro en Windows puede ser un tema de broma, conocidas las limitaciones en cuanto a cuestiones de seguridad de dicho sistema operativo. Además el servidor Apache en su versión Windows esta considerado como "de calidad beta", por lo que no es recomendable su uso en ese sistema más allá que para un test del sistema. Así pues, si realmente necesitas un servidor web opensource y además seguro, considera pasarte a la versión GNU/Linux.

13.2 – Instalando el servidor web apache

Para Windows podemos descargar directamente la última versión de Apache en formato autoextraíble (exe), o en .msi (En estos momentos está la versión 1.3.19 en http://httpd.apache.org/dist/binaries/win32/apache_1.3.19-win32-no_src-r2.msi, véisite <http://httpd.apache.org> para ver cual es la versión más reciente). Para que Windows reconozca estos ficheros deberá tener instalado el Windows Installer. El programa de instalación es exactamente igual al resto de los programas Windows, es decir, le preguntará donde quiere instalar el servidor, el tipo de instalación, etc.

Una vez se ha terminado la instalación, como siempre, deberemos configurar un par de cosas. Para ello editamos el archivo PATH_APACHE/conf/httpd.conf y buscamos y editamos las propiedades ServerName, Port y Listen . En la primera ponemos el nombre de nuestra maquina, en la segunda, el puerto SSL, normalmente el 443, y en la segunda (indica donde esperará peticiones el servidor), ponemos dos valores, el puerto 443 para peticiones seguras, y el 80 para peticiones normales (o los valores que queramos usar).

```
...  
ServerName javahispano
```

```
...  
Port 443
```

```
...  
Listen 80  
Listen 443
```

```
...  
Ahora podemos arrancar el servidor y probar que funciona con los dos puertos con las direcciones
```

```
http://127.0.0.1:80/  
http://127.0.0.1:443/
```

En ambos casos debemos obtener la página de bienvenida de Apache. A partir de este punto (también odemos haerlo al final del proceso) es cuando podemos instalar los módulos de Apache que Queramos, para Tomcat, para php, etc, puesto que nuestro servicio SSL será un módulo más.

13.3 – CREANDO NUESTRO CERTIFICADO

Una vez hemos obtenido la versión de mod_ssl que corresponde a nuestro Apache de la dirección o ftp de mod_ssl (<ftp://ftp.modssl.org/contrib/>) (cada fichero indica la versión de Apache a la que corresponde, así como la versión de mod_ssl y la versión de openssl que incluye) los descomprimos en un directorio temporal.

Importante: Podemos ver que en su versión binaria Windows, mod_ssl incluye ya openssl. No hay que buscarlo y bajarlo de ningún sitio. En la carpeta donde lo haeis descomprimido, encontrareis un directorio llamado openssl que incluye las librerías dll y archivos necesarios. Copiamos entonces los ficheros ssleay32.dll y libeay32.dll al directorio System32 de nuestro Windows, ya sea c:\windows\system32 para Windows 9x/ME o c:\winnt\system32 para Windows NT y Windows 2000. Ahora necesitamos un fichero de configuración de openssl, si no os viene con el mod_ssl, lo podeis descargar de aquí mismo. Lo descargais y lo poneis en la misma carpeta donde esta el programa openssl.exe (algo así como directorio_descrompersion\openssl\bin\). Después de eso podeis hacer doble click sobre el fichero openssl.exe. Vereis que os aparece una ventana de línea de comandos (la típica negra con letras blancas) con el PROMPT openssl>. Es el interfaz de openssl para crear nuestro certificado. Así que lo vamos creando con los siguientes comandos:

NOTA: es importante que el certificado tenga el mismo nombre que nuestra máquina, en nuestro caso "javahispano", pero deberias cambiar ese nombre por el hallais puesto al configurar Apache.

```
req -config openssl.cnf -new -out javahispano.csr
// aqui vamos contestando las preguntas típicas (nombre, organizacion,
etc.)
```

```
rsa -in privkey.pem -out javahispano.key
x509 -in javahispano.csr -out javahispano.cert -req -signkey
javahispano.key -days 365
```

```
//aqui podemos cambiar la opcion "days" por otro valor.
```

```
//este indica que nuestro certificado caduca en un año.
```

```
quit //para cerrar el programa openssl
```

Ya tenemos nuestro certificado (aunque no tiene valor), así que copiamos los ficheros javahispano.cert y javahispano.key a algún sitio donde los encuentre Apache. Lo ideal será crear un subdirectorio llamado ssl dentro de la carpeta PATH_APACHE/conf/ y meterlos allí.

13.4 – CONFIGURANDO APACHE CON MOD_SSL

Lo primero que hacemos es copiar 'TODOS' (leer el parrafo siguiente antes de hacerlo) los ficheros que venían con nuestro paquete mod_ssl en el directorio donde tenemos instalado Apache. Algunos se sobrescribirán, pero no importa, aunque luego quizás tengamos que retocar algunos parametros de nuestro servidor. En las versiones modernas el mod_ssl incluye un fichero conf/httpd.conf , por lo que lo indicado al principio de los puetos y el nombre del servidor se hace necesario de nuevo. Mala suerte, pero es importante hacerlo al principio para saber que el servidor Apache funciona correctamente antes de instalar SSL. Algunos otros parametros que tendremos que reconfigurar son ROOT con el directorio de la instalación del servidor Apache (nombrado en otras partes de este articulo como PATH_APACHE), teniendo en cuenta que hay que usar / en lugar de \, el directorio de usuarios UserDir (comentarlo para no usarlo en Windows), el de documentos DocumentRoot y un poco más abajo Directory el valor algo así como PATH_APACHE/htdocs , y por supuesto lo referente a la configuración de otros modulos, como por ejemplo Tomcat (carga de mod_jk y inclusion de su configuración). No he probado a no copiar este fichero (lo copié antes de saber las consecuencias) y cambiar las cosas necesarias manualmente, pero en fin, si alguien lo hace que me cuente si funciona. Yo os recomiendo que lo intentéis antes así!, aunque el *desastre* no es irreparable tampoco hay porque pegarse un tiro en el pie, ¿no?. Ahora volvemos a editar el fichero PATH_APACHE/conf/httpd.conf para añadirle unas cuantas cosas. En la sección LoadModule añadimos (o simplemente descomentamos, porque seguramente ya esta puesto):

```
...
LoadModule ssl_module modules/ApacheModuleSSL.dll
# o según corresponda, uno de los dos
LoadModule ssl_module modules/ApacheModuleSSL.so
...
```

Y al final del fichero añadimos:

```
# ver http://www.modssl.org/docs/2.4/ssl_reference.html
# para más información.
```

```
SSLMutex sem
SSLRandomSeed startup builtin
SSLSessionCache none
SSLLog logs/ssl.log
SSLLogLevel info
<Virtualhost javahispano:443>
SSLEngine on
SSLCertificateFile conf/ssl/javahispano.cert
SSLCertificateKeyFile conf/ssl/javahispano.key
</Virtualhost>
```

Ya esta todo. Fácil, ¿no?.

13.5 – COMPROBANDO QUE FUNCIONA

Simplemente dirigimos nuestro navegador a la dirección segura de nuestro servidor, en mi caso <https://127.0.0.1/>

Y recibiremos la siguiente ventana informándonos de la entrada en la "zona segura" y de que el certificado no está aprobado por ningún organismo oficial, por lo que podría no ser válido. Podemos examinar el certificado pulsando sobre el tercer botón, algo así como "mostrar certificado". Y nos saldrá la información que introducimos al crearlo.

14 - EJEMPLO DE CONFIGURACION PARA SERVIDOR APACHE+PHP4+SSL+DOMINIOS VIRTUALES

```
#####  
# Ejemplo de configuracion para servidor Apache + PHP4 + SSL + Dominios Virtuales  
#####  
#####  
# Normativas:  
# Servidor Apache Compilado: /usr/local/apache/  
# Servidor Apache Fuentes: /usr/src/apache/  
# Ficheros de Configuracion: /etc/httpd/  
# Ficheros de Log: /var/log/httpd  
# Dominio1: /var/log/httpd/www.midominio.com/  
# Dominio2: /var/log/httpd/www.otrodominio.com/  
# ...  
# WebSites:  
# Dominio1: /home/httpd/www.midominio.com/  
# Dominio2: /home/httpd/www.otrodominio.com/  
# ...  
# Estructura WebSite:  
# Dominio1: /home/httpd/www.dominio.com/  
# Paginas Web: /home/httpd/www.dominio.com/html/  
# Ejecutables CGI: /home/httpd/www.dominio.com/cgi-bin/  
#  
# IMPORTANTE: Sustituir 127.0.0.1 por la IP real del servidor  
#####  
### Section 1: Global Environment  
#  
# The directives in this section affect the overall operation of Apache,  
# such as the number of concurrent requests it can handle or where it  
# can find its configuration files.  
#  
#  
# ServerType is either inetd, or standalone. Inetd mode is only supported on  
# Unix platforms.  
#  
ServerType standalone  
#  
# ServerRoot: The top of the directory tree under which the server's  
# configuration, error, and log files are kept.  
#  
# NOTE! If you intend to place this on an NFS (or otherwise network)  
# mounted filesystem then please read the LockFile documentation  
# (available at <URL:http://www.apache.org/docs/mod/core.html#lockfile>);  
# you will save yourself a lot of trouble.  
#  
# Do NOT add a slash at the end of the directory path.  
#  
ServerRoot /usr/local/apache  
#  
# The LockFile directive sets the path to the lockfile used when Apache  
# is compiled with either USE_FCNTL_SERIALIZED_ACCEPT or  
# USE_FLOCK_SERIALIZED_ACCEPT. This directive should normally be left at  
# its default value. The main reason for changing it is if the logs  
# directory is NFS mounted, since the lockfile MUST BE STORED ON A LOCAL  
# DISK. The PID of the main server process is automatically appended to  
# the filename.  
#  
#LockFile logs/httpd.lock
```

```
#
# PidFile: The file in which the server should record its process
# identification number when it starts.
#
PidFile logs/httpd.pid

#
# ScoreBoardFile: File used to store internal server process information.
# Not all architectures require this. But if yours does (you'll know because
# this file will be created when you run Apache) then you *must* ensure that
# no two invocations of Apache share the same scoreboard file.
#
# ScoreBoardFile logs/httpd.scoreboard

#
# In the standard configuration, the server will process this file,
# srm.conf, and access.conf in that order. The latter two files are
# now distributed empty, as it is recommended that all directives
# be kept in a single file for simplicity. The commented-out values
# below are the built-in defaults. You can have the server ignore
# these files altogether by using "/dev/null" (for Unix) or
# "nul" (for Win32) for the arguments to the directives.
#
#ResourceConfig conf/srm.conf
#AccessConfig conf/access.conf

#
# Timeout: The number of seconds before receives and sends time out.
#
Timeout 300

#
# KeepAlive: Whether or not to allow persistent connections (more than
# one request per connection). Set to "Off" to deactivate.
#
KeepAlive On

#
# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We recommend you leave this number high, for maximum performance.
#
MaxKeepAliveRequests 100

#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.
#
KeepAliveTimeout 15

#
# Server-pool size regulation. Rather than making you guess how many
# server processes you need, Apache dynamically adapts to the load it
# sees --- that is, it tries to maintain enough server processes to
# handle the current load, plus a few spare servers to handle transient
# load spikes (e.g., multiple simultaneous requests from a single
# Netscape browser).
#
# It does this by periodically checking how many servers are waiting
# for a request. If there are fewer than MinSpareServers, it creates
# a new spare. If there are more than MaxSpareServers, some of the
# spares die off. The default values are probably OK for most sites.
#
```

MinSpareServers 5
MaxSpareServers 10

Number of servers to start initially --- should be a reasonable ballpark
figure.

StartServers 5

Limit on total number of servers running, i.e., limit on the number
of clients who can simultaneously connect --- if this limit is ever
reached, clients will be LOCKED OUT, so it should NOT BE SET TOO LOW.
It is intended mainly as a brake to keep a runaway server from taking
the system with it as it spirals down...

MaxClients 250

MaxRequestsPerChild: the number of requests each child process is
allowed to process before the child dies. The child will exit so
as to avoid problems after prolonged use when Apache (and maybe the
libraries it uses) leak memory or other resources. On most systems, this
isn't really needed, but a few (such as Solaris) do have notable leaks
in the libraries.

MaxRequestsPerChild 30

Listen: Allows you to bind Apache to specific IP addresses and/or
ports, in addition to the default. See also the <VirtualHost>
directive.

Listen 443
Listen 80

BindAddress: You can support virtual hosts with this option. This directive
is used to tell the server which IP address to listen to. It can either
contain "*", an IP address, or a fully qualified Internet domain name.
See also the <VirtualHost> and Listen directives.

#BindAddress *

Dynamic Shared Object (DSO) Support

To be able to use the functionality of a module which was built as a DSO you
have to place corresponding `LoadModule' lines at this location so the
directives contained in it are actually available _before_ they are used.
Please read the file README.DSO in the Apache 1.3 distribution for more
details about the DSO mechanism and run `httpd -l' for the list of already
built-in (statically linked and thus always available) modules in your httpd
binary.

Note: The order in which modules are loaded is important. Don't change
the order below without expert advice.

Example:
LoadModule foo_module libexec/mod_foo.so

ExtendedStatus controls whether Apache will generate "full" status
information (ExtendedStatus On) or just basic information (ExtendedStatus

```
# Off) when the "server-status" handler is called. The default is Off.
#
#ExtendedStatus On

### Section 2: 'Main' server configuration
#
# The directives in this section set up the values used by the 'main'
# server, which responds to any requests that aren't handled by a
# <VirtualHost> definition. These values also provide defaults for
# any <VirtualHost> containers you may define later in the file.
#
# All of these directives may appear inside <VirtualHost> containers,
# in which case these default settings will be overridden for the
# virtual host being defined.
#

#
# If your ServerType directive (set earlier in the 'Global Environment'
# section) is set to "inetd", the next few directives don't have any
# effect since their settings are defined by the inetd configuration.
# Skip ahead to the ServerAdmin directive.
#

#
# Port: The port to which the standalone server listens. For
# ports < 1023, you will need httpd to be run as root initially.
#
Port 443

#
# If you wish httpd to run as a different user or group, you must run
# httpd as root initially and it will switch.
#
# User/Group: The name (or #number) of the user/group to run httpd as.
# . On SCO (ODT 3) use "User nouser" and "Group nogroup".
# . On HP-UX you may not be able to use shared memory as nobody, and the
# suggested workaround is to create a user www and use that user.
# NOTE that some kernels refuse to setgid(Group) or semctl(IPC_SET)
# when the value of (unsigned)Group is above 60000;
# don't use Group nobody on these systems!
#
User nobody
Group nobody

#
# ServerAdmin: Your address, where problems with the server should be
# e-mailed. This address appears on some server-generated pages, such
# as error documents.
#
ServerAdmin webmaster@midominio.com

#
# ServerName allows you to set a host name which is sent back to clients for
# your server if it's different than the one the program would get (i.e., use
# "www" instead of the host's real name).
#
# Note: You cannot just invent host names and hope they work. The name you
# define here must be a valid DNS name for your host. If you don't understand
# this, ask your network administrator.
# If your host doesn't have a registered DNS name, enter its IP address here.
# You will have to access it by its address (e.g., http://123.45.67.89/)
# anyway, and this will make redirections work in a sensible way.
#
```

ServerName www.midominio.com

DocumentRoot: The directory out of which you will serve your
documents. By default, all requests are taken from this directory, but
symbolic links and aliases may be used to point to other locations.

DocumentRoot "/home/httpd/www.midominio.com/html"

Each directory to which Apache has access, can be configured with respect
to which services and features are allowed and/or disabled in that
directory (and its subdirectories).

First, we configure the "default" to be a very restrictive set of
permissions.

<Directory />
 Options FollowSymLinks Indexes Includes
 AllowOverride None
</Directory>

DirectoryIndex: Name of the file or files to use as a pre-written HTML
directory index. Separate multiple entries with spaces.

DirectoryIndex index.html index.phtml index.htm index.php3 index.php home.html home.htm

AccessFileName: The name of the file to look for in each directory
for access control information.

AccessFileName .htaccess

The following lines prevent .htaccess files from being viewed by
Web clients. Since .htaccess files often contain authorization
information, access is disallowed for security reasons. Comment
these lines out if you want Web visitors to see the contents of
.htaccess files. If you change the AccessFileName directive above,
be sure to make the corresponding changes here.

<Files .htaccess>
 Order allow,deny
 Deny from all
</Files>

CacheNegotiatedDocs: By default, Apache sends "Pragma: no-cache" with each
document that was negotiated on the basis of content. This asks proxy
servers not to cache the document. Uncommenting the following line disables
this behavior, and proxies will be allowed to cache the documents.

CacheNegotiatedDocs

UseCanonicalName: (new for 1.3) With this setting turned on, whenever
Apache needs to construct a self-referencing URL (a URL that refers back
to the server the response is coming from) it will use ServerName and
Port to form a "canonical" name. With this setting off, Apache will
use the hostname:port that the client supplied, when possible. This
also affects SERVER_NAME and SERVER_PORT in CGI scripts.

UseCanonicalName On

```
#
# TypesConfig describes where the mime.types file (or equivalent) is
# to be found.
#
TypesConfig conf/mime.types

#
# DefaultType is the default MIME type the server will use for a document
# if it cannot otherwise determine one, such as from filename extensions.
# If your server contains mostly text or HTML documents, "text/plain" is
# a good value. If most of your content is binary, such as applications
# or images, you may want to use "application/octet-stream" instead to
# keep browsers from trying to display binary files as though they are
# text.
#
DefaultType text/plain

#
# The mod_mime_magic module allows the server to use various hints from the
# contents of the file itself to determine its type. The MIMEMagicFile
# directive tells the module where the hint definitions are located.
# mod_mime_magic is not part of the default server (you have to add
# it yourself with a LoadModule [see the DSO paragraph in the 'Global
# Environment' section], or recompile the server and include mod_mime_magic
# as part of the configuration), so it's enclosed in an <IfModule> container.
# This means that the MIMEMagicFile directive will only be processed if the
# module is part of the server.
#
<IfModule mod_mime_magic.c>
    MIMEMagicFile conf/magic
</IfModule>

#
# HostnameLookups: Log the names of clients or just their IP addresses
# e.g., www.apache.org (on) or 204.62.129.132 (off).
# The default is off because it'd be overall better for the net if people
# had to knowingly turn this feature on, since enabling it means that
# each client request will result in AT LEAST one lookup request to the
# nameserver.
#
HostnameLookups off
<Files ~ "\.(html|cgi)$">
    HostnameLookups on
</Files>

#
# ErrorLog: The location of the error log file.
# If you do not specify an ErrorLog directive within a <VirtualHost>
# container, error messages relating to that virtual host will be
# logged here. If you *do* define an error logfile for a <VirtualHost>
# container, that host's errors will be logged there and not here.
#
ErrorLog logs/www.midominio.com/error.log

#
# LogLevel: Control the number of messages logged to the error_log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
#
LogLevel warn

#
# The following directives define some format nicknames for use with
```

```
# a CustomLog directive (see below).
#
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

#
# The location and format of the access logfile (Common Logfile Format).
# If you do not define any access logfiles within a <VirtualHost>
# container, they will be logged here. Contrariwise, if you *do*
# define per-<VirtualHost> access logfiles, transactions will be
# logged therein and *not* in this file.
#
CustomLog logs/www.midominio.com/access.log combined

#
# Optionally add a line containing the server version and virtual host
# name to server-generated pages (error documents, FTP directory listings,
# mod_status and mod_info output etc., but not CGI generated documents).
# Set to "EMail" to also include a mailto: link to the ServerAdmin.
# Set to one of: On | Off | EMail
#
ServerSignature On

#
# Aliases: Add here as many aliases as you need (with no limit). The format is
# Alias fakename realname
#
# Note that if you include a trailing / on fakename then the server will
# require it to be present in the URL. So "/icons" isn't aliased in this
# example, only "/icons/".
#
Alias /icons/ /usr/local/apache/icons/

<Directory icons/>
  Options Indexes MultiViews
  AllowOverride None
  Order allow,deny
  Allow from all
</Directory>

<Directory htdocs/>
  Order allow,deny
  Allow from all
</Directory>

#
# ScriptAlias: This controls which directories contain server scripts.
# ScriptAliases are essentially the same as Aliases, except that
# documents in the realname directory are treated as applications and
# run by the server when requested rather than as documents sent to the client.
# The same rules about trailing "/" apply to ScriptAlias directives as to
# Alias.
#
ScriptAlias /cgi-bin/ /home/httpd/www.midominio.com/cgi-bin/

#
# "/usr/local/apache/cgi-bin" should be changed to whatever your ScriptAliased
# CGI directory exists, if you have that configured.
#
<Directory /home/httpd/www.midominio.com/cgi-bin>
  AllowOverride None
  Options None
```

```
    Order allow,deny
    Allow from all
</Directory>

#
# FancyIndexing is whether you want fancy directory indexing or standard
#
IndexOptions FancyIndexing

#
# AddIcon* directives tell the server which icon to show for different
# files or filename extensions. These are only displayed for
# FancyIndexed directories.
#
AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip

AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
AddIconByType (SND,/icons/sound2.gif) audio/*
AddIconByType (VID,/icons/movie.gif) video/*

AddIcon /icons/binary.gif .bin .exe
AddIcon /icons/binhex.gif .hqx
AddIcon /icons/tar.gif .tar
AddIcon /icons/world2.gif .wrl .wrl.gz .vrml .vrm .iv
AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
AddIcon /icons/a.gif .ps .ai .eps
AddIcon /icons/layout.gif .html .shtml .htm .pdf
AddIcon /icons/text.gif .txt
AddIcon /icons/c.gif .c
AddIcon /icons/p.gif .pl .py
AddIcon /icons/f.gif .for
AddIcon /icons/dvi.gif .dvi
AddIcon /icons/uuencoded.gif .uu
AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl
AddIcon /icons/tex.gif .tex
AddIcon /icons/bomb.gif core

AddIcon /icons/back.gif ..
AddIcon /icons/hand.right.gif README
AddIcon /icons/folder.gif ^^DIRECTORY^^
AddIcon /icons/blank.gif ^^BLANKICON^^

#
# DefaultIcon is which icon to show for files which do not have an icon
# explicitly set.
#
DefaultIcon /icons/unknown.gif

#
# AddDescription allows you to place a short description after a file in
# server-generated indexes. These are only displayed for FancyIndexed
# directories.
# Format: AddDescription "description" filename
#
#AddDescription "GZIP compressed document" .gz
#AddDescription "tar archive" .tar
#AddDescription "GZIP compressed tar archive" .tgz

#
# ReadmeName is the name of the README file the server will look for by
# default, and append to directory listings.
#
# HeaderName is the name of a file which should be prepended to
```

```
# directory indexes.
#
# The server will first look for name.html and include it if found.
# If name.html doesn't exist, the server will then look for name.txt
# and include it as plaintext if found.
#
ReadmeName README
HeaderName HEADER

#
# IndexIgnore is a set of filenames which directory indexing should ignore
# and not include in the listing. Shell-style wildcarding is permitted.
#
IndexIgnore .??* *~*# HEADER* README* RCS CVS *,v *,t

#
# AddEncoding allows you to have certain browsers (Mosaic/X 2.1+) uncompress
# information on the fly. Note: Not all browsers support this.
# Despite the name similarity, the following Add* directives have nothing
# to do with the FancyIndexing customization directives above.
#
AddEncoding x-compress Z
AddEncoding x-gzip gz

#
# AddLanguage allows you to specify the language of a document. You can
# then use content negotiation to give a browser a file in a language
# it can understand. Note that the suffix does not have to be the same
# as the language keyword --- those with documents in Polish (whose
# net-standard language code is pl) may wish to use "AddLanguage pl .po"
# to avoid the ambiguity with the common suffix for perl scripts.
#
AddLanguage en .en
AddLanguage fr .fr
AddLanguage de .de
AddLanguage da .da
AddLanguage el .el
AddLanguage it .it

#
# LanguagePriority allows you to give precedence to some languages
# in case of a tie during content negotiation.
# Just list the languages in decreasing order of preference.
#
LanguagePriority en fr de

#
# AddType allows you to tweak mime.types without actually editing it, or to
# make certain files to be certain types.

# For PHP 4.x, use:
#
AddType application/x-httpd-php .php
AddType application/x-httpd-php .php3
AddType application/x-httpd-php .php4
AddType application/x-httpd-php .phtml
AddType application/x-httpd-php-source .phps

#
# AddHandler allows you to map certain file extensions to "handlers",
# actions unrelated to filetype. These can be either built into the server
# or added with the Action command (see below)
#
# If you want to use server side includes, or CGI outside
```

```
# ScriptAliased directories, uncomment the following lines.
#
# To use CGI scripts:
#
#AddHandler cgi-script .cgi

#
# To use server-parsed HTML files
#
AddType text/html .shtml
AddHandler server-parsed .shtml
AddHandler server-parsed .html

#
# Some MIME-types for downloading Certificates and CRLs
#
AddType application/x-x509-ca-cert .crt
AddType application/x-pkcs7-crl .crl

#
# Uncomment the following line to enable Apache's send-asis HTTP file
# feature
#
#AddHandler send-as-is asis

#
# If you wish to use server-parsed imagemap files, use
#
#AddHandler imap-file map

#
# To enable type maps, you might want to use
#
#AddHandler type-map var

#
# Action lets you define media types that will execute a script whenever
# a matching file is called. This eliminates the need for repeated URL
# pathnames for oft-used CGI file processors.
# Format: Action media/type /cgi-script/location
# Format: Action handler-name /cgi-script/location
#

#
# MetaDir: specifies the name of the directory in which Apache can find
# meta information files. These files contain additional HTTP headers
# to include when sending the document
#
#MetaDir .web

#
# MetaSuffix: specifies the file name suffix for the file containing the
# meta information.
#
#MetaSuffix .meta

#
# Customizable error response (Apache style)
# these come in three flavors
#
# 1) plain text
#ErrorDocument 500 "The server made a boo boo."
# n.b. the (") marks it as text, it does not get output
#
```

```

# 2) local redirects
#ErrorDocument 404 /missing.html
# to redirect to local URL /missing.html
#ErrorDocument 404 /cgi-bin/missing_handler.pl
# N.B.: You can redirect to a script or a document using server-side-includes.
#
# 3) external redirects
#ErrorDocument 402 http://some.other_server.com/subscription_info.html
# N.B.: Many of the environment variables associated with the original
# request will *not* be available to such a script.

#
# The following directives modify normal HTTP response behavior.
# The first directive disables keepalive for Netscape 2.x and browsers that
# spoof it. There are known problems with these browser implementations.
# The second directive is for Microsoft Internet Explorer 4.0b2
# which has a broken HTTP/1.1 implementation and does not properly
# support keepalive when it is used on 301 or 302 (redirect) responses.
#
BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\0b2;" nokeepalive downgrade-1.0 force-response-1.0

#
# The following directive disables HTTP/1.1 responses to browsers which
# are in violation of the HTTP/1.0 spec by not being able to grok a
# basic 1.1 response.
#
BrowserMatch "RealPlayer 4\0" force-response-1.0
BrowserMatch "Java/1\0" force-response-1.0
BrowserMatch "JDK/1\0" force-response-1.0

#
# Allow server status reports, with the URL of http://servername/server-status
# Change the ".your_domain.com" to match your domain to enable.
#
<Location /server-status>
    SetHandler server-status
    Order deny,allow
    Deny from all
</Location>

#
# Allow remote server configuration reports, with the URL of
# http://servername/server-info (requires that mod_info.c be loaded).
# Change the ".your_domain.com" to match your domain to enable.
#
<Location /server-info>
    SetHandler server-info
    Order deny,allow
    Deny from all
</Location>

#
# There have been reports of people trying to abuse an old bug from pre-1.1
# days. This bug involved a CGI script distributed as a part of Apache.
# By uncommenting these lines you can redirect these attacks to a logging
# script on phf.apache.org. Or, you can record them yourself, using the script
# support/phf_abuse_log.cgi.
#
#<Location /cgi-bin/phf*>
#    Deny from all
#    ErrorDocument 403 http://phf.apache.org/phf_abuse_log.cgi
#</Location>

```

```

### Parametros SSL genericos
SSLVerifyClient 0
SSLVerifyDepth 10
SSLCertificateKeyFile /etc/httpd/new.cert.key
SSLCertificateFile /etc/httpd/new.cert.cert

#####
# Note: The following directives are only required if session
# cacheing is enabled (the default from 1.17). To disable
# cacheing, make sure the following is set in apache_ssl.c
#
#define CACHE_SESSIONS      FALSE
SSLCacheServerPath /usr/local/apache/bin/gcache
SSLCacheServerPort /tmp/ssl.fictional.co.cache.socket
SSLSessionCacheTimeout 300
# end conditional section

### Section 3: Virtual Hosts
#
# VirtualHost: If you want to maintain multiple domains/hostnames on your
# machine you can setup VirtualHost containers for them.
# Please see the documentation at <URL:http://www.apache.org/docs/vhosts/>
# for further details before you try to setup virtual hosts.
# You may use the command line option '-S' to verify your virtual host
# configuration.
#
# If you want to use name-based virtual hosts you need to define at
# least one IP address (and port number) for them.
#
NameVirtualHost 127.0.0.1:80
NameVirtualHost 127.0.0.1:443

<VirtualHost 127.0.0.1:80>
  Redirect / http://www.midominio.com/
  ServerAdmin webmaster@midominio.com
  SSLDisable
  Port 80
  TransferLog /dev/null
</VirtualHost>

#####
#####
<VirtualHost www.midominio.com:80>
  ServerAdmin webmaster@midominio.com
  SSLDisable
  Port 80
  DocumentRoot /home/httpd/www.midominio.com/html
  ServerName www.midominio.com
  ErrorLog logs/www.midominio.com/error.log
  CustomLog logs/www.midominio.com/access.log combined
  ScriptAlias /cgi-bin/ /home/httpd/www.midominio.com/cgi-bin/
</VirtualHost>

#####
#####
### Section 4: Modulos ...
#
# CheckSpelling
#
#CheckSpelling On
#####
#Ejemplo de configuracion para servidor Apache + PHP4 + SSL +Dominios
#Virtuales

```