UNIVERSIDAD NACIONAL DE LA MATANZA
Departamento de Ingeniería e Investigaciones Tecnológicas

Seguridad y Calidad en Aplicaciones Web

Certificados Digitales

Referente de Cátedra: Walter R. Ureta
Plantel Docente: Pablo Pomar, Walter R. Ureta

Acrónimos

- **BER(Basic Encoding Rules), CER (Canonical Encoding Rules), DER (Distinguished Encoding Rules)**: certificado en forma binaria encodeados según la **ITU-T X.690** (Con base en ASN.1)
- **CRT (Certificate):** En referencia a certificados CER o DER.
- **CRL ("Certificate Revocation List"):** Lista de certificados revocados
- **PEM (Privacy-enhanced Electronic Mail):** RFC1421/22/23/24, usualmente aplicado en un certificado DER encodeado en Base64 contenido entre las líneas: "-----*BEGIN CERTIFICATE*-----" y "-----*END CERTIFICATE*-----"
- **JKS (Java Key Store):** Formato de repositorios de claves-certificados
- **PKCS (Public-Key Cryptography Standards):** Estándares definidos por RSA Security para el manejo de información con algoritmos asimétricos. La siguiente tabla enumera y sintetiza estos estándares

Acrónimo	Contenido	Referencias y detalles
PKCS#1	Estándar criptográfico RSA	RFC 3447. Define el formato del cifrado RSA.
PKCS#2	*Obsoleto*	Cifrado de resúmenes. Incorporado en PKCS#1
PKCS#3	Estándar de intercambio de claves Diffie-Hellman	-
PKCS#4	*Obsoleto*	Sintaxis de claves. Incorporado en PKCS#1
PKCS#5	Estándar de cifrado basado en contraseñas	Padding. RFC 2898 y PBKDF2
PKCS#6	Estándar de sintaxis de certificados extendidos	Extensiones X.509 v1. (No utilizado en V3)
PKCS#7	Estándar sobre la sintaxis del mensaje criptográfico	RFC 2315. Firmar y cifrar mensajes PKI (.p7b , .p7c)
PKCS#8	Estándar sobre la sintaxis de la información de clave privada	RFC 5208
PKCS#9	Tipos de atributos seleccionados	-
PKCS#10	Estándar de solicitud de certificación	RFC 2986. CSR, solicitud de firma de clave publica
PKCS#11	Interfaz de dispositivo criptográfico ("CryptographicTokenInterface" ocryptoki)	Define un API genérico de acceso a dispositivos criptográficos
PKCS#12	Estándar de sintaxis de intercambio de información personal	Formato de repositorio de claves/certificados. (.p12, .pfx)
PKCS#13	Estándar de criptografía de curva elíptica	-
PKCS#14	Generación de número pseudo-aleatorios	-
PKCS#15	Estándar de formato de información de dispositivo criptográfico	-

OpenSSL

OpenSSL es un proyecto de software libre basado en **SSLeay**, desarrollado por Eric Young y Tim Hudson. El mismo esta distribuido bajo licencia Apache, consta de herramientas y bibliotecas criptográficas que asisten a implementaciones de sistemas de seguridad como SSL, TLS y SSH.

Este software también puede ser utilizado para generar certificados en servidores como Apache y Tomcat.

Generar Clave RSA

Comando para crear una clave RSA

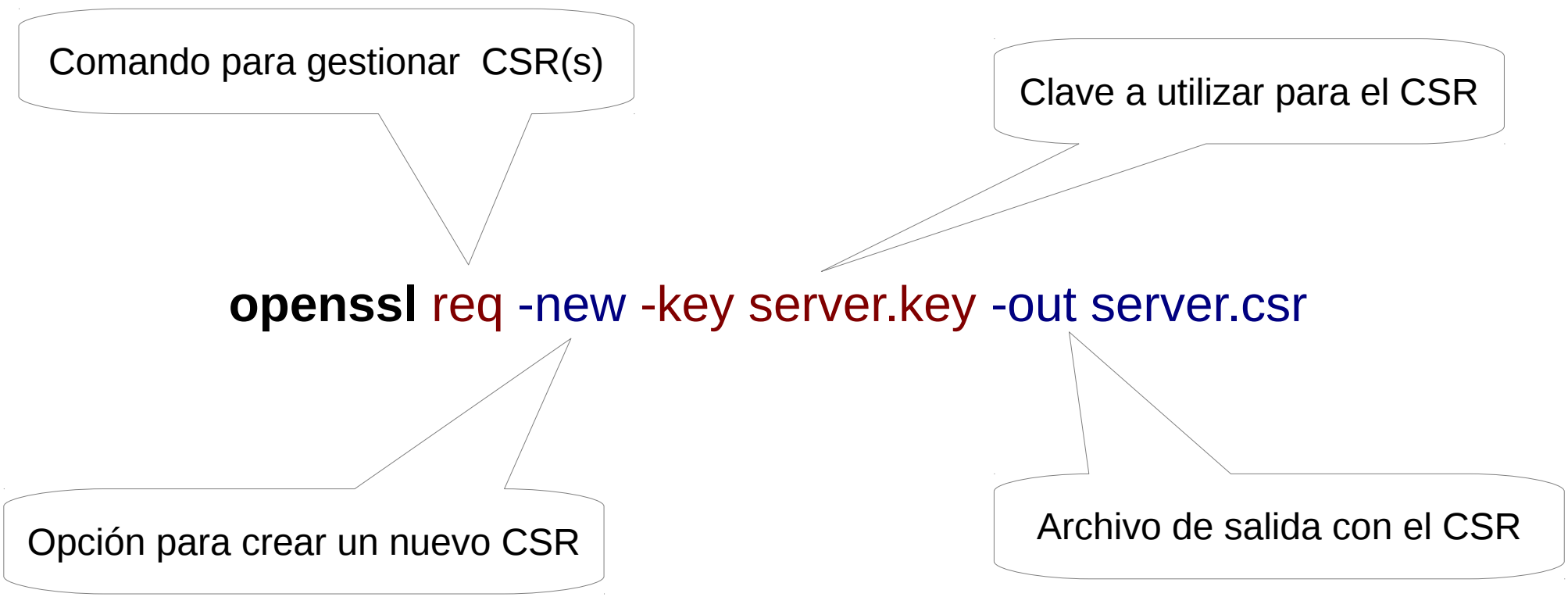
Archivo de salida

openssl genrsa -des3 -out server.key 1024

Modo de cifrado. Opciones: aes256,camelia192,seed ....

Cantidad de la clave a generar

Generar CSR (Certificate Signing Request)



Generar un certificado “Self-Signed”

Comando para gestionar certificados x509

Cantidad de días de validez

Clave a utilizar para firmar el certificado

openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt

Indicar que la entrada es un CSR

Archivo a firmar. Entrada (CSR)

Certificado firmado. Archivo de salida.

Esta es una alternativa a comprar o solicitar la firma de un tercero confiable (Ej. Verisign)

Remover la contraseña de una clave RSA

Archivo con la clave original

Copia de la clave que mantendrá la contraseña

```
cp server.key server.key.orig
openssl rsa -in server.key.orig -out server.key
```

Comando para gestionar claves RSA

Clave RSA con contraseña. Archivo de entrada

Copia de la clave sin contraseña. Archivo de Salida

SSL en Apache Server (httpd)

Instalar la clave privada y su certificado

```
cp server.crt /usr/local/apache/conf/ssl.crt/
cp server.key /usr/local/apache/conf/ssl.key/
```

Configurar SSL en el Virtual-Host (httpd-vhosts.conf)

```
SSLEngine on
SSLCertificateFile /usr/local/apache/conf/ssl.crt/server.crt
SSLCertificateKeyFile /usr/local/apache/conf/ssl.key/server.key
SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-shutdown
CustomLog logs/ssl_request_log \
  "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
```

Reiniciar el servidor Apache

SSL en NginX

Instalar la clave privada y su certificado

```
cp server.crt /etc/nginx/ssl/
cp server.key /etc/nginx/ssl/
```

Configurar SSL en el archivo de configuración (Ej. nginx.conf)

```
server {
        listen 443;
        server_name [SERVER_NAME];
        root /usr/share/nginx/www;
        index index.html index.htm;
        ssl on;
        ssl_certificate /etc/nginx/ssl/server.crt;
        ssl_certificate_key /etc/nginx/ssl/server.key;
}
```

Reiniciar el servidor NginX

TLS en Apache Tomcat

Apache tomcat dispone de dos implementaciones diferentes para utilizar SSL en la comunicación. Estas son:

- Implementación **JSSE**, provista por Java desde la versión 1.4
- Implementación **APR**, utiliza OpenSSL

Tomcat dispone de una herramienta para manejo de certificados y repositorios denominada "**keytool**", adicionalmente cabe destacar que por defecto se maneja el formato **"jks"** (Java Key Store) en lugar de **"PKCS#12"**.

TLS en Apache Tomcat

Opcionalmente, el conector a utilizar puede ser seleccionado mediante la configuración del conector correspondiente en el archivo server.xml. A continuación se muestra como realizar esta selección

```
<!-- Define a blocking Java SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector protocol="org.apache.coyote.http11.Http11Protocol"
        port="8443" .../>

<!-- Define a non-blocking Java SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector protocol="org.apache.coyote.http11.Http11NioProtocol"
        port="8443" .../>

<!-- Define a APR SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector protocol="org.apache.coyote.http11.Http11AprProtocol"
        port="8443" .../>
```

Creación de repositorio en PKCS12

Gestión de archivos PKCS12

Certificado de entrada

Archivo PKCS12 de salida

openssl pkcs12 -export -in server.crt -inkey server.key -out server.p12 -name server12

Indica la generación del archivo en pkcs12 como salida

Clave privada de entrada

Nombre de la entrada (Certificado-Clave) en el PKCS12

TLS en Apache Tomcat

A fin de habilitar el conector correspondiente debemos editar el archivo **server.xml** con una entrada que configura el mismo indicando los valores específicos de los parámetros y el repositorio de certificados a utilizar.
El siguiente es un ejemplo para la implementación por defecto de Java (JSSE)

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
        maxThreads="150" scheme="https" secure="true"
        clientAuth="false" sslProtocol="TLS"
        keystoreFile="${user.home}/server.p12"
       keystorePass="1234" keystoreType= "PKCS12"
       keyAlias="server12" />
```

Algunas versiones pueden sufrir un error de padding al leer la clave cifrada, se sugiere utilizar un repositorio JKS.

TLS en Apache Tomcat

Como alternativa podemos generar un repositorio JKS e importar la información del PKCS#12 de los pasos previos

keytool -genkey -alias foo -keystore "./keystore.jks"
keytool -delete -alias foo -keystore "./keystore.jks"
Keytool -importkeystore -destkeystore keystore.jks -srckeystore keystore.p12 -srcstoretype pkcs12 -alias server12
keytool -changealias -keystore keystore.jks -alias server12 -destalias tomcat
keytool -list -keystore keystore.jks

Crear JKS vacio

Importar clave y certificado

Listar contenido del JKS

Cambiar alias (Opcional)

A continuación se muestra la configuración de ejemplo:

```
<Connector
        protocol="org.apache.coyote.http11.Http11NioProtocol"
        port="8443" maxThreads="200"
        scheme="https" secure="true" SSLEnabled="true"
        keystoreFile="${user.home}/keystore.jks"  keystorePass="12345678"
        keyAlias="tomcat" keyPass="1234"
        clientAuth="false" sslProtocol="TLS"/>
```

Generar JKS con clave y certificado auto-firmado

Podemos crear un repositorio JKS con una nueva clave y certificado autofirmado de la siguiente manera.

Ante la pregunta "Nombre y apellido" se debe ingresar el nombre del dominio a utilizar. Ej: www.server.net

TLS en Apache Tomcat

Apache Tomcat también puede manejar la auténticamente de los clientes vía certificados para ello se debe tener un repositorio de certificados de confianza y habilitar el modo en cuestión.
El siguiente es un ejemplo de la configuración para este caso.

```
<Connector
    protocol="org.apache.coyote.http11.Http11NioProtocol"
    port="8443" maxThreads="200"
    scheme="https" secure="true" SSLEnabled="true"
    keystoreFile="${user.home}/keystore.jks" keystorePass="12345678"
    keyAlias="tomcat" keyPass="1234"
    truststoreFile="${user.home}/trustKeystore.jks" truststorePass="12345678"
    clientAuth="true" sslProtocol="TLS"/>
```

SSL en Apache Tomcat

Este es el ejemplo correspondiente a la implementación de APR, donde no utilizamos un repositorio sino los archivos del certificado y la clave en formato PEM.

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector
        protocol="HTTP/1.1"
        port="8443" maxThreads="200"
        scheme="https" secure="true" SSLEnabled="true"
        SSLCertificateFile="/usr/local/ssl/server.crt"
        SSLCertificateKeyFile="/usr/local/ssl/server.key"
        SSLVerifyClient="optional" SSLProtocol="TLSv1"/>
```

Certificados digitales en Navegadores Web

A fin de manejar las implementaciones de SSL/TLS los navegadores web actuales necesitan disponer de un repositorio de certificados y funcionalidades para administrarlos.
En términos generales encontraremos las siguientes categorías para la administración de los mismos:

- **Certificados Propios** - Se utilizan para identificar al usuario frente a aplicaciones web que requieran este nivel de autenticación.
- **Certificados de Servidores** - Se utilizan para que el browser considere a un servidor externo como "*Confiable*". Algunos browser permiten crear "*Excepciones*" para cuando un servidor no dispone de un certificado en su base, de esta forma se descarga el mismo y se utiliza de forma temporal.
- **Certificados de Autoridades** - Se utilizan para validar la firma de un CA sobre un certificado de un servidor y de esta forma considerarlo "*Confiable*" en base al reconocimiento del CA.

Certificados digitales en Mozilla Firefox

Para acceder a la administración de certificados en Mozilla Firefox debe seguir los siguientes pasos

- Seleccionar el item "**Editar**" en la barra de menú
- Seleccione la opción de "**Preferencias**"
- Ir a la sección de "**Avanzado**"
- Elegir la solapa "**Encriptación**"
- Dentro de la sección de certificados encontrará el botón "**Ver Certificados**", el mismo permite acceder a las siguientes categorías con posibilidad de gestión:
  - Sus Certificados *(PKCS#12)*
  - Personas *(DER,CER,CRT,PEM)*
  - Servidores *(DER,CER,CRT,PEM)*
  - Autoridades *(DER,CER,CRT,PEM)*
  - Otros *(Dispositivos de seguridad)*

Certificados digitales en Google Chrome

Para acceder a la administración de certificados en Google Chrome debe seguir los siguientes pasos

- Haz clic en el menú de Chrome situado en la barra de herramientas del navegador.
- Selecciona **Configuración**.
- Haz clic en la opción para **mostrar la configuración avanzada**.
- Dentro de la sección de "**HTTPS/SSL**" encontrará el botón "**Administrar Certificados**", el mismo permite acceder a las siguientes categorías con posibilidad de gestión:
  - Sus Certificados *(PKCS#12)*
  - Servidores *(DER,CRT,PEM,PKCS#7/p7c)*
  - Autoridades *(DER,CRT,PEM,PKCS#7/p7c)*
  - Otros *(Dispositivos de seguridad)*

Referencias

Lista de Software

OpenSSL: http://www.openssl.org/
Servidor Apache: http://httpd.apache.org/
Apache Tomcat: http://tomcat.apache.org/
Mozilla Firefox: http://www.mozilla.org/firefox/
Google Chrome: http://www.google.com/Chrome/
Soporte del Browser: https://cc.dcsec.uni-hannover.de/

Algunas Autoridades de certificación

CA Cert Org: http://www.cacert.org/
Verisign: http://www.verisign.com/
DigiCert: http://www.digicert.com/
Certisur: https://www.certisur.com