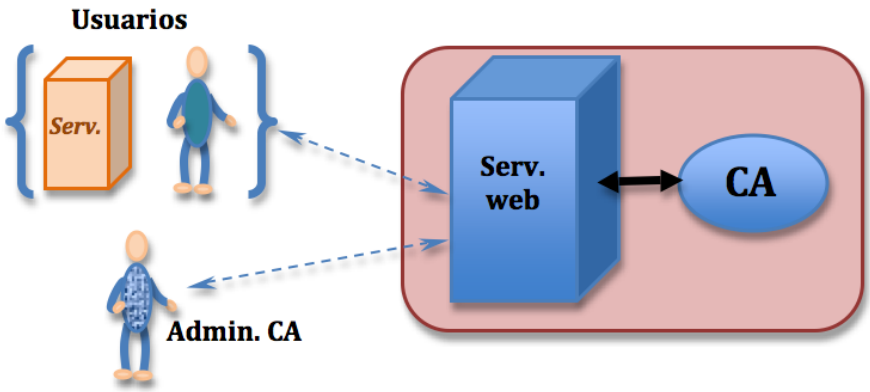
Práctica : PKI

Objetivo

Consolidar los conceptos teóricos relacionados con certificados digitales. La parte fundamental de esta práctica será poner en funcionamiento una PKI simple compuesta por una autoridad de certificación (CA) que emita y verifique certificados[1].

Tareas

- Poner en marcha una PKI[2] que emita/revoque/verifique certificados para usuarios (personas y servidores) de una red privada.
- Diseñar una interfaz web para dicha PKI, con la descripción de todas las funcionalidades necesarias para interactuar con ella.



Documentación a entregar:

- Elaborar un **guión detallado** con todos los pasos necesarios comentados, incluyendo ficheros de configuración, capturas de tramas, capturas de pantalla [y scripts] que indique como :
  - Crear la autoridad de certificación
  - Generar certificados para usuarios (personas y sitios web)
  - Revocar certificados
  - Verificar certificados

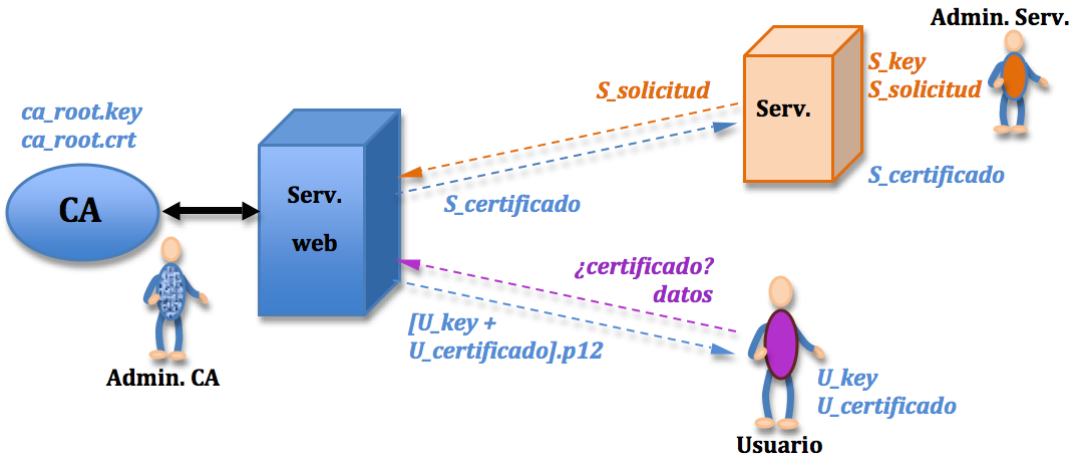
[1] En una práctica posterior se hará uso de estos certificados con la puesta en marcha de sitios web seguros.

[2] En una máquina virtual VMware con S.O. Linux Debian 8.1

- **Diseñar una aplicación web** que sirva de interfaz con la CA. A esta aplicación tienen acceso los usuarios para solicitar y verificar certificados. También tendrá acceso el administrador de la CA para gestionar todas las tareas de la CA (acceso restringido). No hay que implementarla
- Contabilizar el tiempo invertido

Ejercicio extra Crear una CA subordinada de la CA raíz.

Descripción del servicio ofrecido por la CA



- Puede generar dos [tres] tipos de certificados:
  - Personal:
    El usuario solicita un certificado para él y/o su dirección de correo electrónico. Proporciona sus **datos personales** que la CA verificará; si supera esta verificación remitirá al usuario un fichero protegido con contraseña que contendrá el par *<clave privada, certificado>*
  - Para un servidor:
    El administrador de un servidor solicita un certificado. Proporciona a la CA una **solicitud de certificado** (que incluye una clave pública y datos adicionales sobre el servidor). La CA verificará los datos y firmará el certificado que hará llegar al solicitante
  - [Para otra CA subordinada (actuando como una CA raíz). Emitirá un certificado para la CA subordinada, válido para expedir nuevos certificados]
- La emisión de certificados no podrá ser automática. En esta práctica, el administrador de la CA asumirá el papel de la Autoridad de Registro, verificando los datos aportados por los solicitantes, y concediendo o denegando la emisión del certificado
- Publicar su certificado, hacerlo accesible a todos los clientes de la red
- Revocar certificados
- Gestionar los números de serie de los certificados emitidos y revocados
- Gestionar la lista de certificados revocados (CRL), mantenida y distribuida desde la CA
- Gestionar un servidor OCSP para verificar certificados emitidos por la CA

Soporte: Crear una infraestructura de clave pública (PKI)

OpenSSL es una colección de bibliotecas de funciones y comandos para criptografía y comunicaciones bajo el protocolo SSL/TLS. Proporciona todos los comandos necesarios para poner en marcha una PKI. En esta actividad se utilizará una aplicación sencilla (`ca`) disponible en OpenSSL que facilita la gestión de una base de datos para los certificados emitidos y revocados por la CA, y que trabaja con un fichero de configuración donde se recoge información para emitir y revocar certificados.

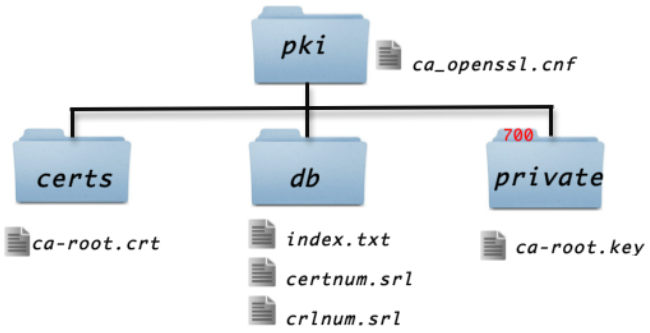
Creación de la Autoridad de Certificación Raíz (CA root)

1. Crear la estructura de directorios de la CA raíz.

   Ejemplo:

   En un directorio principal crear (al menos) tres subdirectorios[3] que contendrán:

   - Certificados emitidos por la CA
   - Ficheros con números de serie para certificados emitidos y CRLs; y el fichero índice para la base de datos
   - Clave privada de la CA (recomendable directorio con permisos restringidos)



2. Inicializar los ficheros de números de serie de certificados emitidos y revocados (contienen el siguiente número de serie en hexadecimal). Crear el fichero índice (vacío) para la base de datos

   - Para crear un fichero vacío:
     `# touch nombre-fichero`
   - Para inicializar un fichero con un valor hexadecimal aleatorio:
     `# openssl rand -hex 8 > nombre-fichero`

[3] Los nombres adoptados, se indicarán en el fichero de configuración

3. Completar el fichero de configuración *ca_openssl.cnf*
   Asignar valores coherentes a todos los campos que figuren entre *«...*
   ...»

4. Generar la clave de la CA raíz y su certificado[4]

```
# openssl req -new -nodes -keyout clave-privada-CA-root \
      -out solicitud-CA-root -config fichero-configuración
# openssl ca -selfsign \
      -in solicitud-CA-root -out certificado-CA-root \
      -config fichero-configuración -extensions ca_ext
```

5. Generar una CRL vacía

```
# openssl ca -gencrl -out crl-CA-root.pem \
      -config fichero-configuración
```

6. Generar la clave y el certificado para un servidor OCSP

```
# openssl req -new -nodes -out solicitud-ca-ocsp \
      -keyout clave-privada-ca-ocsp \
      [-subj datos-solicitud]
# openssl ca -in solicitud-ca-ocsp -out certificado-ca-ocsp \
      -config fichero-configuración -extensions ocsp_ext
```

Emisión de un certificado

```
# openssl req -new -nodes -out solicitud-usuario \
      -keyout clave-privada-usuario \
      [-subj datos-solicitud]
# openssl ca -in solicitud-usuario -out certificado-usuario \
      -config fichero-configuración -extensions {server_ext, client_ext}
```

Revocación de un certificado

```
# openssl ca -revoke certificado-servidor \
      -crl_reason keyCompromise -config fichero-configuración

   reason: {keyCompromise, affiliationChanged, unspecified,
      CACompromise, superseded, removeFromCRL,...}
En función de las políticas de la PKI, actualizar CRL
```

[4]Sustituir los nombres en cursiva por la **ruta absoluta** al fichero correspondiente

Validación de un certificado (*OCSP responder)*

- Poner en marcha un servidor OCSP (*responder*) con `OpenSSL`. Utiliza la base de datos creada por la CA, y espera las peticiones de los clientes en el puerto indicado

```
[# rm log.txt ]
# openssl ocsp -port num_puerto -index indice_base_datos \
      -CA certificado-CA-root -rsigner certificado-ca-ocsp \
      -rkey clave-privada-ca-ocsp -text -out log.txt
```

- Preguntar al servidor OCSP (actuando como cliente OCSP)

```
# openssl ocsp -CAfile certificado-CA-root \
      -issuer certificado-CA-root \
      -url url_oscp_servidor:num_puerto \
      -cert certificado-servidor -resp_text
```

Exportar certificado y clave privada

- Exportar el certificado y la clave privada de un usuario a un fichero con formato PKCS#12 (se establece una contraseña de cifrado a conocer por CA y usuario)

```
openssl pkcs12 -export -out fichero.p12 -in certificado.pem -inkey clave-privada.
```

- Extraer certificado y clave [5]

```
openssl pkcs12 -in fichero.p12 -out certificado.pem
```

Opciones

- Para obtener una solicitud de certificado de forma no interactiva, se pasan todos los datos solicitados por medio de la opción `-subj` *"/campo0=valor0/campo1=valor1/.../campoN=valorN/"*. Por ejemplo:

```
-subj "/C=ES/ST=GI/L=SS/O=SRDSI/OU=LABS-SRDSI/
      CN=ocsp.srdsi.lab/emailAddress=webmaster@srdsi.lab/"
```

- El formato de los ficheros generados por OpenSSL es PEM. Son ficheros de texto, pero no son legibles. Para comprobar el contenido de estos ficheros:

```
# openssl {req, x509, rsa, crl} -in filename.pem [-noout] -text
```

- Algunos ficheros deben distribuirse en formato DER (formato binario) como, por ejemplo, las CRLs. Para cambiar de formato PEM a DER:

```
# openssl crl -inform PEM -in fichero.pem \
      -outform DER -out fichero.der
```

[5] También se puede importar el fichero .p12 directamente al navegador.

- Generar certificados multi-dominio (y *wildcard*).
    - Crear un fichero con la extensión de nombres alternativos
      `# echo "subjectAltName = DNS:*.srdsi.lab , DNS:srdsi.lab" > file.ext`
    - Al generar el certificado incluir la opción `-extfile file.ext`

Referencias

- «*Openssl cookbook*» I. Ristic (2015)
- http://pki-tutorial.readthedocs.org/en/latest/index.html

ca_openssl.cnf

```
[default]
name          = …nombre-CA…
domain_suffix = …dominio, por ejemplo srdsi.lab…
aia_url       = http://$name.$domain_suffix/$name.crt
crl_url       = http://$name.$domain_suffix/$name.crl
ocsp_url      = http://ocsp.$domain_suffix:…num_puerto…
default_ca    = CA_default   # Sección por defecto para CA
name_opt      = utf8,esc_ctrl,multiline,lname,align

[ca_dn]
countryName             = …nombre_país (2 letras)…
stateOrProvinceName     = …nombre_provincia (2 letras)…
localityName            = …nombre_localidad…
organizationName        = …nombre_organización…
organizationalUnitName  = …nombre_departamento…
commonName              = …nombre_identificador_certificado…


[CA_default]
home             = …directorio_principal_estructura…
db               = …directorio_base_datos…
privado          = …directorio_datos_privados…
database         = $home/$db/…fich_indice_db.txt…
serial           = $home/$db/…fich_num_serie_certificados.srl…
crlnumber        = $home/$db/…fich_num_serie_crl.srl…
certificate      = $home/$name.crt
private_key      = $home/$privado/$name.key
crl              = $home/$db/$name.crl         # (DER format)
RANDFILE         = $home/$privado/.rand
new_certs_dir    = $home/…directorio_nuevos_certificados…
unique_subject   = no
copy_extensions  = none
default_days     = 365
default_crl_days = 30
default_md       = default                      # sha256
policy           = policy_c_o_match

# Política para la CA
[policy_c_o_match]
countryName             = match
stateOrProvinceName     = optional
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

[req]
default_bits       = 2048                      # 4096
encrypt_key        = yes
default_md         = default                   # sha256
utf8               = yes
string_mask        = utf8only
prompt             = no
distinguished_name = ca_dn
req_extensions     = ca_ext
```

```
[ca_ext]
basicConstraints     = critical,CA:true
keyUsage             = critical,keyCertSign,cRLSign
subjectKeyIdentifier = hash
[crl_info]
URI.0     = $crl_url

[issuer_info]
caIssuers;URI.0    = $aia_url
OCSP;URI.0         = $ocsp_url

[name_constraints]
permitted;DNS.0=…nombres-dominio, srdsi.lab…
permitted;DNS.1=…otros-nombres-dominio, srdsi.labs…
excluded;IP.0=0.0.0.0/0.0.0.0
excluded;IP.1=0:0:0:0:0:0:0:0/0:0:0:0:0:0:0:0

[ocsp_ext]
authorityKeyIdentifier  = keyid:always
basicConstraints        = critical,CA:false
extendedKeyUsage        = OCSPSigning
keyUsage                = critical,digitalSignature
subjectKeyIdentifier    = hash

[server_ext]
authorityInfoAccess     = @issuer_info
authorityKeyIdentifier  = keyid:always
basicConstraints        = critical,CA:false
crlDistributionPoints   = @crl_info
extendedKeyUsage        = clientAuth,serverAuth
keyUsage                = critical,digitalSignature,keyEncipherment
subjectKeyIdentifier    = hash

[client_ext]
authorityInfoAccess     = @issuer_info
authorityKeyIdentifier  = keyid:always
basicConstraints        = critical,CA:false
crlDistributionPoints   = @crl_info
extendedKeyUsage        = clientAuth
keyUsage                = critical,digitalSignature
subjectKeyIdentifier    = hash
```