Seguridad en Redes

Práctica 2.3. Certificados digitales y modelos de confianza

OpenSSL

Creación de una CA

Lo primero que tenemos que hacer es crear una CA que, posteriormente, firmará los certificados de usuario que se soliciten. Es conveniente revisar el fichero de configuración por defecto (`/usr/lib/ssl/openssl.cnf`).
Aunque normalmente habría que adaptar la configuración a nuestras necesidades, en este caso, usaremos directamente la configuración por defecto, para lo cual, hay que crear algunos ficheros y directorios:

```
$ mkdir demoCA
$ mkdir demoCA/newcerts
$ mkdir demoCA/private
$ touch demoCA/index.txt
$ echo 01 > demoCA/serial
$ echo 01 > demoCA/crlnumber
```

- **Entrega:** Si decidiésemos utilizar unos directorios distintos a los usados anteriormente, a la hora de trabajar con el certificado y la clave privada de la CA, ¿nos daría algún error? De ser así, ¿qué deberíamos hacer para poder usar esos directorios?

Para crear un certificado raíz autofirmado a partir de una nueva clave privada, se usa el comando `req` con las opciones `-x509` y `-new`:

```
$ openssl req -x509 -new -days 3650 -keyout
demoCA/private/cakey.pem -out demoCA/cacert.pem
```

Con la opción `-newkey` en lugar de `-new`, se pueden especificar los parámetros de la clave, por ejemplo `-newkey rsa:1024`. Con la opción `-key` en lugar de `-keyout`, se puede utilizar una clave ya existente.

- Crea un certificado raíz autofirmado para la CA (con contraseña “seguridad”). Rellenar los campos *Country* con SP, *State name* con Madrid, *Locality name* también con Madrid, *Organization name* con *SER Corporation* y ***Common Name* (CN) con CA**.

  - **Entrega:** Copia el fichero con el certificado de la CA y el fichero con la clave privada de la CA. ¿En qué directorio o directorios se han guardado estos ficheros? ¿De qué tipo y de qué tamaño es la clave generada por defecto?

Creación de solicitudes de firma de certificado

Para crear una CSR (*Certificate Signing Request*), se usa el comando `req` con la opción `-new`:

```
$ openssl req -new -keyout userkey.pem -out usercsr.pem
```

También se pueden usar las opciones `-newkey` y `-key`.

- Crea varias solicitudes de firma de certificado (con contraseña "seguridad"). Rellenar los campos *Country*, *State name*, *Locality name* y *Organization name* con los mismos valores usados para la CA. **A *Common Name* (CN) se le debe asignar un nombre único por solicitud, por ejemplo, usuario1, usuario2,** etc.

La solicitud, que va firmada con la clave privada del solicitante, puede verse con:

```
$ openssl req -in usercsr.pem -noout –text
```

- **Entrega:** ¿Qué es lo que contiene la solicitud de firma del certificado? ¿Qué algoritmo de *hash* se ha usado para generar la firma digital? ¿Con qué clave se ha firmado la solicitud del certificado?

Se puede verificar la firma con:

```
$ openssl req -verify -in usercsr.pem
```

Creación y verificación de certificados

Para firmar una CSR con la CA por defecto y generar un certificado, se usa el comando `ca`:

```
$ openssl ca -in usercsr.pem -out usercert.pem
```

Antes de firmar el certificado se comprueba la firma de la solicitud y se pide confirmación.

Para verificar un certificado, se usa el comando `verify`, indicando con la opción `-CAfile` el fichero con los certificados de las CAs en las que se confía (en este caso, solamente una):

```
$ openssl verify -CAfile demoCA/cacert.pem usercert.pem
```

- Firma las solicitudes creadas en el ejercicio anterior para generar certificados. Verifica los certificados.

Consulta y manipulación de certificados

Para extraer información del certificado (aunque normalmente se puede ver en el fichero del certificado), se usa el comando `x509`. Por ejemplo:

```
$ openssl x509 -in usercert.pem -noout -text
```

```
$ openssl x509 -in usercert.pem -noout –pubkey
```

- Examina los certificados creados (CA y usuarios). Observa las diferencias entre un certificado de CA y un certificado de usuario.

  - **Entrega:** Copia la información extraída de varios ficheros con certificados de usuario. ¿Qué datos aparecen vinculados al certificado? ¿Qué versión de certificado se está usando? ¿Cómo podemos diferenciar un certificado de una CA de un certificado de usuario?

El formato PEM (*Privacy Enhaced Mail*) usa el formato binario DER (*Distinguish Encoding Rules*) pero codificado en Base64 y entre cabeceras ASCII, de modo que es apropiado para transferencias en modo texto entre sistemas. Para indicar el formato DER de entrada o salida, se usa el comando `x509` con las opciones `-inform` y `-outform`, respectivamente. Por ejemplo:

```
$ openssl x509 -in usercert.pem -out usercert.der -outform DER
$ openssl x509 -in usercert.der -inform DER -out usercert.pem
```

Para exportar un certificado, junto con su clave, a formato PKCS12 (usado en muchos navegadores y clientes de correo) se usa el comando `pkcs12`:

```
$ openssl pkcs12 -export -in usercert.pem -inkey userkey.pem -
out usercert.p12
```

- Cambia los certificados de usuario creados a formato DER y PKCS12.

  - **Entrega:** Copia un fichero con un certificado en formato PKCS12.

Revocación de certificados

Para revocar un certificado, se usa el comando `ca` con la opción `-revoke`:

```
$ openssl ca -revoke usercert.pem
```

Para generar una nueva CRL (*Certificate Revocation List*), se usa el comando `ca` con la opción `-gencrl`:

```
$ openssl ca -gencrl -out crl.pem
```

Para examinar la CRL, se usa el comando `crl` con la opción `-text`:

```
$ openssl crl -in crl.pem -noout -text
```

Para verificar la CRL, se usa el comando `crl` con la opción `-CAfile`:

```
$ openssl crl -CAfile demoCA/cacert.pem -in crl.pem
```

Para verificar un certificado comprobando que no está en la CRL, se usa el comando `verify` con la opción `-crl_check`, indicando con la opción `-CRLfile` el fichero con la CRL:

```
$ openssl verify -crl_check -CAfile demoCA/cacert.pem -CRLfile
crl.pem usercert.pem
```

Si esto no funciona (por tratarse de una versión anterior), hay que crear un fichero con los certificados de las CAs y las CRLs e indicarlo con la opción `-CAfile`:

```
$ cat demoCA/cacert.pem crl.pem > cacrl.pem
$ openssl verify -CAfile cacrl.pem -crl_check usercert.pem
```

- Revoca uno de los certificados de usuario creados, crea una CRL y verifica si el certificado ha sido revocado comprobando el contenido de la CRL y usando el comando `verify` con la opción `-crl_check`.

  - **Entrega:** Copia la información contenida en el fichero de la CRL. ¿Hay algún certificado revocado? ¿Quién lo ha revocado? ¿Quién ha creado la CRL? ¿Cómo se sabe, al examinar la CRL, qué certificado ha sido revocado? ¿Por qué se confía en la validez de la CRL?

GnuPG

Firma de claves (*Web of trust*)

Para firmar una clave una vez que se ha verificado su validez, se usa el comando `--sign-key`:

```
$ gpg2 --sign-key <id>
```

Si no se especifica cuál es la clave firmante, por defecto, se firmará siempre con la primera clave que aparece en el anillo.
Para firmar con una clave concreta, se usa la opción `--local-user`:

```
$ gpg2 --sign-key --local-user <id1> <id2>
```

Se firmará la clave con identificador `id2` con la clave con identificador `id1`.

Para asignar un nivel de confianza en el propietario de la clave, se usa el comando `--edit-key` con el subcomando `trust`:

```
$ gpg2 --edit-key <id> trust quit
```

Para mostrar la validez de las claves, hay que utilizar la opción `show-uid-validity`:

```
$ gpg2 --list-options show-uid-validity --list-keys
```

Para recalcular la validez de las claves y mostrar un resumen de la red de confianza, se usa el comando `--check-trustdb`:

```
$ gpg2 --check-trustdb
```

- Crea una clave con nombre “claveA” (con contraseña “seguridad”). Descarga el fichero `pubkeysSER.gpg` del Campus Virtual. Importar las claves públicas contenidas en este fichero. Las claves importadas han sido creadas con la contraseña “Segurid@d”.

- Firma B con A y C con B. Establece la confianza en (el propietario de) B a *total* (4) y la confianza en C a *desconocida* (1). Comprueba la validez de las claves.

  - **Entrega:** Copia la salida del comando que muestra la validez de las claves (`gpg2 --list-options show-uid-validity --list-keys`) y del que muestra el resumen de la red de confianza (`gpg2 --check-trustdb`). Explica el contenido de la red de confianza ¿Cuál será la validez de C?

- Cambia la confianza en B a *dudosa* (3) y comprueba de nuevo la validez de las claves.

  - **Entrega:** Copia la salida del comando que muestra la validez de las claves y del que muestra el resumen de la red de confianza. Explicar el contenido de la red de confianza ¿Cuál será ahora la validez de C?

- Firma las claves D y E con A (para que sean válidas), establece su confianza a *dudosa* (3) y firma con ellas la clave C. Comprueba otra vez la validez de las claves.

  - **Entrega:** Copia la salida del comando que muestra la validez de las claves y del que muestra el resumen de la red de confianza. Explicar el contenido de la red de confianza ¿Cuál será ahora la validez de C? Dibuja la red de confianza siguiendo el esquema en forma de árbol visto en las transparencias del Módulo 2.

Opcional

Si decides hacer alguno de estos apartados opcionales añade la explicación de lo que hace cada comando junto con los resultados obtenidos a la memoria de la práctica.

S/MIME con OpenSSL

Consulta los comandos `smime` (para S/MIME 3.0 con PKCS#7) y `cms` (S/MIME 3.1 con *Cryptographic Message Syntax*, CMS) y practica con ellos.
Ejemplos de uso (con el comando `cms` sería similar):

```
$ openssl smime -encrypt -in message.txt -out ciphered.txt
usercert.pem
$ openssl smime -decrypt -recip usercert.pem -inkey
userkey.pem -in ciphered.txt
$ openssl smime -sign -signer usercert.pem -inkey userkey.pem
```

```
-in message.txt -out signed.txt
$ openssl smime -verify -signer usercert.pem -CAfile
demoCA/cacert.pem -in signed.txt
```

S/MIME con GnuPG

Consulta la página de manual del comando `gpgsm` y practica con él.

Comparación del rendimiento

Compara el rendimiento de distintas operaciones usando el comando `speed` de `openssl`:

```
$openssl speed algorithm
```

- Cifrar con clave simétrica
- Hacer un hash
- firmar vs. verificar con RSA
- firmar vs. verificar con DSA