1.- Comprobando autenticidad de claves

Firmar claves públicas de otros

Si estoy seguro de que cierta clave pública pertenece a B, puedo firmar dicha clave pública con mi clave privada. De esta forma tendré la certeza en el futuro de que la clave es de quién dice ser.

Antes de firmar una clave pública debo estar REALMENTE seguro de que pertenece a la persona a la que supuestamente pertenece. Si no estas con una seguridad del 100% NO FIRMES la clave, esto sería engañarte a ti mismo y también a los demás como explicaré después.

Para asegurarte de que una clave pertenece a B hay dos formas:

-1 De forma directa hablas con B en persona (o por teléfono si reconoces su voz) y veis que la clave es buena (esto es un poco aburrido al principio cuando tienes pocas llaves, pero al final no nos escribimos con tanta gente diferente y una vez hecho vale para toda la vida)

-2 De forma indirecta a través de amigos comunes

Para validar la clave de forma directa es necesario que ya sea medio teléfono, o con una tarjeta de visita o como sea que estés seguro de hablar con B comprobéis los fingerprints de las claves. El fingerprint es una versión abreviada de la clave para poder comparar rapidamente.

Para ver el fingerprint haz:
"gpg --fingerprint KeyID" o bien
"gpg --fingerprint direccion@de_correo

una vez comprobados ya puedes validar la clave y para eso hay que firmarla:
gpg --edit-key KeyID
y después puedes hacer unos comandos (haz help) entre ellos "sign"

De esta forma lo que dices es que: "estas completamente seguro de que la clave de B que tienes realmente pertenece a B y que por tanto las firmas de B serán de un usuario validado". Al recibir un mensaje de B verás algo así como: "Good signature from trusted B"

Si no firmas la clave no es demasiado grave, podrás leer los mensajes que te lleguen de B pero te quedará una pequeña duda de que no haya nadie que este suplantando la identidad de B. A pesar de eso podrás comprobar que el mensaje no a sido modificado desde que se envió, porque la firma será buena y pondrá algo así al recibir un mensaje de B: "Good signature from untrusted B".

Lo siguiente que puedes hacer es establecer un nivel de **"trust"** para la clave, es decir, establecer un nivel de confianza de que B sabe lo que esta haciendo con lo del PGP (lo que

hemos hecho antes era establecer un nivel de validity), por ejemplo

- Tú puedes estar seguro de que la clave de B y la mía pertenecen realmente a cada uno de nosotros dos, pero sin embargo que te fíes más de mi en cuanto a usuario de PGP que de B (cosa independiente de que te fíes de que B sea B, este concepto es **validity**)

La idea fundamental es que si tú te fías completamente de mi buen uso del PGP, las claves que yo valide quedan automáticamente validadas para ti, sin necesidad de que establezcas comunicación directa con las otras personas, por ejemplo:

Si en una empresa hay alguien que se dedica meticulosamente a verificar las claves y todos confían en el con un nivel de trust muy alto, a través de él podéis establecer un nivel de confianza con los demás de forma indirecta. Además, en el momento que esta persona revoke una clave y diga que no es fiable todos cambiaran su opinión sobre esa clave (por ejemplo si alguien deja la empresa)

Supongamos que A conoce a B y B a C, y que A quiere enviar un mensaje a C.

Si B ha validado la clave de C y A la de B, y además A se fía de B, entonces puede estar seguro que la clave de C es de quien dice que es.

Para ello, lo que tiene que haber pasado anteriormente es lo siguiente:

-1- A comprueba la clave de B y la firma como hemos dicho antes, pero además una vez firmada exporta sus claves y las publica en Teams, en el grupo de SAD.
Lo que hace entonces es enviar las claves que tiene firmadas por él, pero **NO los niveles de trust** (eso es algo privado y muy subjetivo que a nadie le interesa, por ejemplo B se podría sentir molesto si ve en un servidor que no crees que controle el PGP.... ,
Para enviar las claves: gpg --send-keys --keyserver pgp.mit.edu

-2- B hace lo mismo de antes con C

-3- Cuando A quiere enviar un mensaje a C o comprobar su firma, debe importar la clave publica de C y establecer un nivel de validity de la clave, como a través de B llegas a C, se hará un nivel de validity de C de TRUSTED, es decir que te creerás que C es C porque B lo ha comprobado y tú te fías de B. Si B no hubiera firmado la clave de C entonces lo tendrías como UNTRUSTED.

Fíjate como lo que se establece de esta forma es el nivel de validity, pero el nivel de TRUST de C lo tienes que poner tu de forma personal, como decía antes eso es algo subjetivo y que no se exporta al servidor, **solo uno sabe en quien debe confiar**.

También date cuenta que a C se puede llegar a través de otra gente, quizás de la que te fías parcialmente, si esto es así, generalmente dos personas de las que te fías parcialmente (nivel de trust) hacen como una de la que te fías totalmente. De vez en cuando las personas intermedias pueden ir firmando claves (extendiendo la red), para actualizar el anillo haz:

gpg --update-trustdb
gpg --refresh-keys

Ah y por último para cambiar el nivel de confianza de una clave haz:

gpg --edit-key keyID

y luego el comando trust"

Ejercicio 1a : ¿Cómo firmar un manifiesto toda la clase?

Crear un manifiesto sobre las condiciones de ambientales del aula, tanto de temperatura como de sonido. El escrito debe estar firmado por todos los asistentes y enviado a la profesora para que sólo ella lo pueda leer.

Ejercicio 1b: Valida todas las claves de los alumnos del aula. Para ello debes tener el fingerpirnt firmado de cada uno de tus compañeros.

2.- Generar y convertir claves y certificados con OpenSSL

Usando los comandos expuestos en este artículo y con OpenSSL podemos crear una clave pública y privada para usarlo con ssh o para cifrar y descifrar mensajes, un certificado autofirmado que podremos usar en un servidor de aplicaciones para usar un protocolo seguro y también convertir las claves y certificados a uno de los formatos aceptados por la aplicación que usemos.

Para un uso personal como enviar correos o archivos cifrados o firmados digitalmente usar GnuPG es una buena opción. En Internet los servidores también se aprovechan del uso de criptografía para realizar comunicaciones seguras entre el usuario y el servidor.

Para hacer uso en un servidor de una comunicación https donde los datos viajan cifrados y sin que otras partes salvo el usuario y el servidor puedan acceder a los datos necesitamos un certificado digital. Un certificado es un archivo que contiene la clave pública sirviéndonos para verificar su autenticidad. Un certificado autofirmado es un certificado firmado con la misma clave privada asociada a la clave pública que contiene el certificado. Un certificado autofirmado es suficiente para un entorno de pruebas pero en un servidor para proporcionar confianza a los usuarios deberemos solicitar que una autoridad de certificados que nos firme con su clave nuestro certificado, si el usuario confía en esa autoridad de certificado puede de esta manera confiar en nuestro certificado y clave pública. Varias entidades de registro de dominios o alojamiento web ofrecen la compra de certificados SSL, en el artículo Certificado SSL, de empresa, «wildcard» y de validación extendida comento con un poco más detalle los varios tipos de certificados y algunas opciones donde obtenerlos o comprarlos.

Dependiendo del tipo de certificado que solicitemos y nos entregue la autoridad de certificado el usuario podrá ver que está simplemente accediendo a un servidor con conexión segura, ver los detalles de nuestro certificado y en algunos casos el usuario podrá ver en la

barra de direcciones en verde el nombre de la entidad, que puede darle al usuario más confianza y ver que realmente está accediendo al servidor correcto y no a uno que esté intentando suplantar una identidad. En este último caso la barra de direcciones no tendría en verde el nombre de la entidad, esto es algo que como usuarios debemos comprobar al acceder a determinados sitios de forma segura.

Con la herramienta OpenSSL y los siguientes comandos podemos generar claves y certificados y realizar las conversiones entre formatos que necesitemos. Una vez que disponemos de un certificado y del formato en el que necesitemos podemos hacer uso de él, por ejemplo, en un servidor de páginas web o aplicaciones para proporcionar acceso mediante el protocolo HTTPS y proporcionar seguridad SSL. En otros artículos muestro cómo configurar SSL/TLS en un servidor Tomcat, JBoss, WildFly, Lighttpd, Nginx o Apache, por otro lado usar HTTPS es un requisito para utilizar el protocolo HTTP/2 en un servidor web.

Contenido del artículo

Crear claves y certificados

Crear una clave privada y pública

Para generar un par de claves RSA pública y privada que nos permitan tanto cifrar datos como realizar firmas se emplea el siguiente comando:

```
1 $ openssl genrsa -out localhost.key 8192
2
```

script-1.sh

Para cifrar la clave generada con el algoritmo *aes256* y protegerla por una contraseña se puede emplear el siguiente comando, en realidad al generar la clave indicando la misma opción *-aes256* en el comando anterior la clave se generará cifrada y protegida por una

contraseña. Para cambiar la contraseña es el mismo comando y el segundo comando elimina la contraseña y la descifra:

```
1 $ openssl rsa -aes256 -in localhost.key -out localhost-encrypted.key
2 $ openssl rsa -in localhost-encrypted.key -out localhost.key
```

script-12.sh

El contenido de un archivo de clave privada sin cifrar tiene el siguiente aspecto (los tres puntos son líneas de contenido omitidas).

```
 1 -----BEGIN RSA PRIVATE KEY-----
 2 MIISKAIBAAKCBAEAyMuPWmmTI4IkfoqQ2TWpqo79GEasdxn9McHb4I5Vk4c3rFIR
 3 ExMozoWmCVvKD6Z9lT9aJ69jlwAR4KGownT4mXN0pRN3mSf6kZ854XssB6o63/0e
 4 /D1xTYdcrP7OKMnHyZwcVKPXEz0RhUsUqH7wzIjwsoXDBDOVFM6EJ2RGo+MuTMDX
 5 wMW3X/DvnDlaxes7ZZizrQ0F+hvopFZowZxRjj/RCJRdAbzNhVEyG+2qtaYlRIC3
 6 mq/eYCJUi+jD1jnFr2daC4zFa6Cr7bGoxrXJlNg4iS1hjciaMV13kXXW1lsWOkPd
 7 exctVEb/nuH64cozPhXJoBjR7rDvXPllnhTay3UnVqK7art1iqKZXmN7FGGM2nJv
 8 w7PIYk25T7H3ucRwfT+svUzdooGNSpojKhFoaAvG4X5Z9wZgtTa24pz/T673DH+J
 9 1Hujco+ufAwJ+ZMZDfnN2g49Y2hqrJmU177e+g6vW5xYAK8raYp6RzTYRuxKLaQg
10 mi4ctBsycbxETZUk1kSlOB2ebM6+8lArW8VOtCQGdNvvEfydrTHK3hxrEvW8oFAb
11 j5eU3oWzZL0wTMJ1S/Crbg2eRfCkWPvBdV3e5DUUlqMp2f5urxgOlHmkYEop8gBm
12 ...
13 ...
14 ...
15 ...
16 ...
17 ...
18 sH+8kfyWZP4x+ivVlTrtJMRPJLKAbIJ2CEN16YQtaE9tjCWcz4JjV2b7k79UEjLg
19 mVGXSWo8koxl8MJpSFlYJMAlhiLkoanNzW+u+LPvn2Abc6174u5QD+ihk8c=
20 -----END RSA PRIVATE KEY-----
```

localhost.key

Exportar la clave pública

El archivo generado al crear el par de claves contiene tanto la clave pública como la privada. La privada no se debe distribuir y se debe mantener protegida de forma que solo la conozca su propietario por ejemplo guardándola en una base de datos de KeePassXC como un archivo adjunto de la aplicación KeePassXC. La clave pública es la que se distribuye a otras personas o entidades. Para extraer la clave pública del archivo generado anterior por OpenSSL usamos el siguiente comando:

```
1 $ openssl rsa -in localhost.key -pubout > localhost.pub
2
```

script-9.sh

También se puede obtener la clave pública en formato OpenSSH y una representación gráfica de la huella digital.

```
1 $ ssh-keygen -y -f localhost.key > localhost-openssh.pub
2
```

script-10.sh

```
1 $ ssh-keygen -lv -f localhost-openssh.pub
2 8192 SHA256:mxKfFP7NwwlZsvfxMqG9CQOjJssyOA/v1I67C4j8vwM no comment
3 (RSA)
```

```
+---[RSA 8192]----+
|                 |
|                 |
|         . . .   |
|        . . =    |
|       . S * . o |
|o. E . + * O = + |
|o.o.o + B . X = .|
|  +=o= =     + = |
|   =XO*       o  |
  +----[SHA256]-----+
```

script-15.sh

Obtener la huella digital de la clave pública

La huella digital de una clave pública sirve para comprobar que la clave es la esperada. Son una cadena de números y letras pudiendo estar cada pareja de caracteres separados por ∴.

```
$ openssl dgst -sha256 -c localhost.pub
SHA256(localhost.pub)=
a4:39:88:cf:8b:cf:3c:13:23:c1:e6:f8:10:d3:e5:a4:95:6f:30:d6:11:b1:81:4b:c
e:3d:c0:e5:1c:57:ca:ff
```

script-11.sh

Crear un certificado

Un certificado permite utilizar el protocolo seguro HTTPS en un servidor web y contiene la firma de una tercera parte que valida nuestra clave pública como auténtica. Para que esa tercera parte pueda firmar nuestra clave deberemos generar una petición de firma de certificado y enviársela a la autoridad de certificado que nos lo devolverá firmado. La petición firma de certificado se crea con el siguiente comando:

```
$ openssl req -new -key localhost.key -out localhost.csr

```

script-2.sh

Si no queremos tratar con una autoridad de certificado, ya que cobran por la firma, podemos crear un certificado autofirmado que puede ser suficiente para un entorno de pruebas de un servidor web. El comando para generar el certificado autofirmado es, la opción *-subj* indica la información del sujeto a autenticar:

```
$ openssl req -new -x509 -days 1825 -subj
"/C=ES/ST=Spain/L=/O=/CN=localhost" -key localhost.key -out
localhost.crt
```

script-3.sh

Convertir un certificado a otros formatos

Dependiendo de la autoridad de certificado el certificado puede estar en diferentes formatos, también dependiendo del servidor donde se quiera usar es necesario convertirlo a al formato

adecuado. OpenSSL permite para hacer las conversiones entre formatos DER, PEM y PKCS#12.

Convertir un certificado en formato DER (.crt .cer .der) a PEM

```
1 $ cat localhost.key localhost.crt > localhost.pem
2 $ openssl x509 -in localhost.crt -out localhost.pem
3 $ openssl x509 -inform der -in localhost.cer -out localhost.pem
```

script-4.sh

Verificar un certificado en formato PEM

```
1 $ openssl verify -verbose localhost.pem
```

script-4v.sh

Convertir un certificado en formato PEM a DER

```
1 $ openssl x509 -outform der -in localhost.pem -out localhost.der
2
```

script-5.sh

Convertir un certificado en formato PEM y una clave privada a PKCS#12 (.pfx .p12)

```
1 $ openssl pkcs12 -export -out localhost.p12 -inkey localhost.key -in
2 localhost.crt
```

script-6.sh

Convertir un archivo en formato PKCS#12 (.pfx .p12) que contiene una clave privada y certificado a PEM

```
1 $ openssl pkcs12 -in localhost.p12 -out localhost.pem -nodes
2
```

script-7.sh

Examinar certificados

Examinar un certificado

```
  $ openssl x509 -text -noout -in localhost.crt
1 Certificate:
      Data:
2         Version: 3 (0x2)
          Serial Number:
3             49:e5:32:9b:7f:be:bf:a6:90:14:34:5e:24:53:e8:96:51:8d:74:d4
          Signature Algorithm: sha256WithRSAEncryption
4         Issuer: C = ES, ST = Spain, CN = localhost
          Validity
5             Not Before: Jul 27 10:40:19 2020 GMT
              Not After : Jul 26 10:40:19 2025 GMT
6         Subject: C = ES, ST = Spain, CN = localhost
```

```
          Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (8192 bit)
                Modulus:
                    00:c8:cb:8f:5a:69:93:23:82:24:7e:8a:90:d9:35:
                    a9:aa:8e:fd:18:46:ac:77:19:fd:31:c1:db:e0:8e:
                    ...
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                07:A0:9B:0F:8F:22:9D:AE:CF:F7:46:11:82:B7:7A:0C:43:88:58
:8D
            X509v3 Authority Key Identifier:
                keyid:07:A0:9B:0F:8F:22:9D:AE:CF:F7:46:11:82:B7:7A:0C:43
:88:58:8D

            X509v3 Basic Constraints: critical
                CA:TRUE
    Signature Algorithm: sha256WithRSAEncryption
         1b:9d:d1:a5:1a:2d:23:8c:09:8b:08:6c:fb:49:f6:88:e9:51:
         86:ce:19:53:74:ab:90:01:d9:ab:d1:9e:2b:56:ca:7c:a3:53:
         ...
```

script-13.sh

Examinar el certificado de un servidor web

```
$ openssl s_client -showcerts -connect duckduckgo.com:443
CONNECTED(00000003)
```

```
 4 depth=2 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert
 5 Global Root CA
 6 verify return:1
 7 depth=1 C = US, O = DigiCert Inc, CN = DigiCert SHA2 Secure Server CA
 8 verify return:1
 9 depth=0 C = US, ST = Pennsylvania, L = Paoli, O = "Duck Duck Go, Inc.",
10 CN = *.duckduckgo.com
11 verify return:1
12 ---
13 Certificate chain
14  0 s:C = US, ST = Pennsylvania, L = Paoli, O = "Duck Duck Go, Inc.", CN
15 = *.duckduckgo.com
16    i:C = US, O = DigiCert Inc, CN = DigiCert SHA2 Secure Server CA
17 -----BEGIN CERTIFICATE-----
18 MIIGNTCCBR2gAwIBAgIQAx42ydSKbld6na9pgqdX4zANBgkqhkiG9w0BAQsFADBN
19 MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMScwJQYDVQQDEx5E
20 ...
21 -----END CERTIFICATE-----
22  1 s:C = US, O = DigiCert Inc, CN = DigiCert SHA2 Secure Server CA
23    i:C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert
24 Global Root CA
25 -----BEGIN CERTIFICATE-----
26 MIIElDCCA3ygAwIBAgIQAf2j627KdciIQ4tyS8+8kTANBgkqhkiG9w0BAQsFADBh
27 MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
28 ...
29 -----END CERTIFICATE-----
30 ---
31 Server certificate
32 subject=C = US, ST = Pennsylvania, L = Paoli, O = "Duck Duck Go, Inc.",
   CN = *.duckduckgo.com

   issuer=C = US, O = DigiCert Inc, CN = DigiCert SHA2 Secure Server CA

   ---
   ...
```

script-14.sh

Autoridad de certificación

Los comandos anteriores permiten generar un certificado autofirmado válido para proporcionar una conexión cifrada entre un servidor y un cliente como es el caso de un servidor web y un navegador web. Pero los certificados autofirmados no permiten todas las validaciones de seguridad, el cliente no puede confiar en que realmente se está conectando al nombre del dominio del servidor que el certificado incluye y por ello en el cliente hay que eliminar la validación de comprobación del certificado. Para mayor seguridad y en un entorno de producción se ha de utilizar una autoridad de certificación, es posible crear una autoridad de certificación propia con comandos de OpenSSL.

Referencia:

- OpenSSL
- The Most Common OpenSSL Commands

Ejercicio 2a: Haz un resumen de los comandos OpenSSL si quieres crear:

1. Un par de claves autofirmadas de longitud 2048 en un archivo llamado pkey.pem y protegido con la contraseña vagrant e importar la clave pública al fichero public-pkey.pem
2. Una solicitud de firma de claves (csr)
3. Un certificado autofirmado de 1 año de duración y que se guradará en el fichero selfsignedcert.pem