Comprobación de la integridad con el Hash de un fichero

Probablemente habrás descargado muchas ISO sin tener la seguridad de que estaban bien: si se habían corrompido en la descarga, o bien alguien había hecho un Man In the Middle y te había colocado una ISO que no era …ve a la página de descarga de GPARTED y comprueba qué contienen los ficheros que se llaman CHECKSUM.TXT. Son los hash de las ISO que están disponibles para la descarga.

Lee este artículo para conocer de lo que vamos a tratar en este apartado: las funciones HASH

- Responde a las siguientes preguntas:
  - ¿Por qué se dice que el hash es una función unidireccional?

    Porque de los datos puedes sacar el hash, pero del hash **no** puedes sacar los datos.
    Es decir, es un sistema seguro para comprobar el contenido, sin exponerlo.

  - ¿Por qué se almacenan hash-es de las contraseñas en las bases de datos y no las contraseñas en sí mismas? ¿qué conseguimos con ello?

    Porque de esta manera garantizamos seguridad y privacidad:
    1- **Seguridad:** si de algún modo un usuario malicioso accede a nuestra base de datos, no podrá recoger las contraseñas reales (por lo explicado en el punto anterior), por lo que no podrá acceder a sus cuentas o intentar acceder a otras plataformas utilizando la misma contraseña o similares.
    2- **Privacidad:** la página no conocerá la contraseña del usuario, sólo su Hash, por lo que se respetará la privacidad del usuario (esta privacidad realmente otorga también a su vez una gran seguridad).

  - ¿Estas dos cadenas de texto producirían hashes similares? “Hola” <> “hola”

    Nope, cualquier cambio produce (o debería producir, por ejemplo, MD5 y Sha1 tienen ciertas debilidades) una enorme o total alteración en el resultado del Hash, incluidos los cambios en mayúsculas y minúsculas.

- Crea una carpeta dentro de $HOME/cripto llamada hashing
  mkdir hashing

- Usa el comando wget para descargar la última versión de la ISO para 64 bits de GPARTED en una carpeta Descargas dentro del $HOME

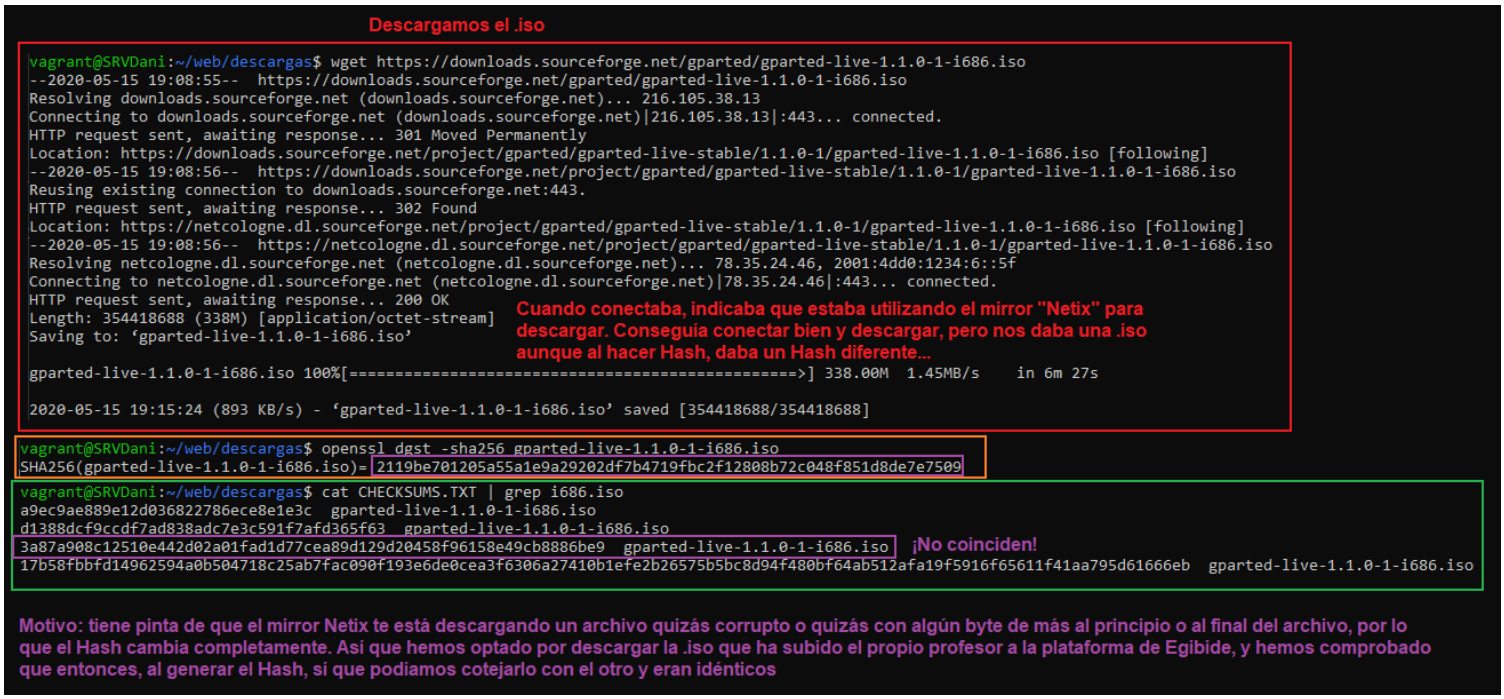
```
vagrant@SRVDani:~/web/descargas$ wget https://downloads.sourceforge.net/gparted/gparted-live-1.1.0-1-i686.iso
```

- Consulta en la web los ficheros donde aparecen los HASH de ese fichero ISO. ¿Qué formatos hay? ¿Cuál es el más robusto?

  En la lista CHECKSUM.TXT (lista de Hash-es de los diferentes archivos) están los formatos:
  - **MD5:** inseguro
  - **SHA1:** débil, inseguro
  - **SHA256:** seguro
  - **SHA512:** muy seguro

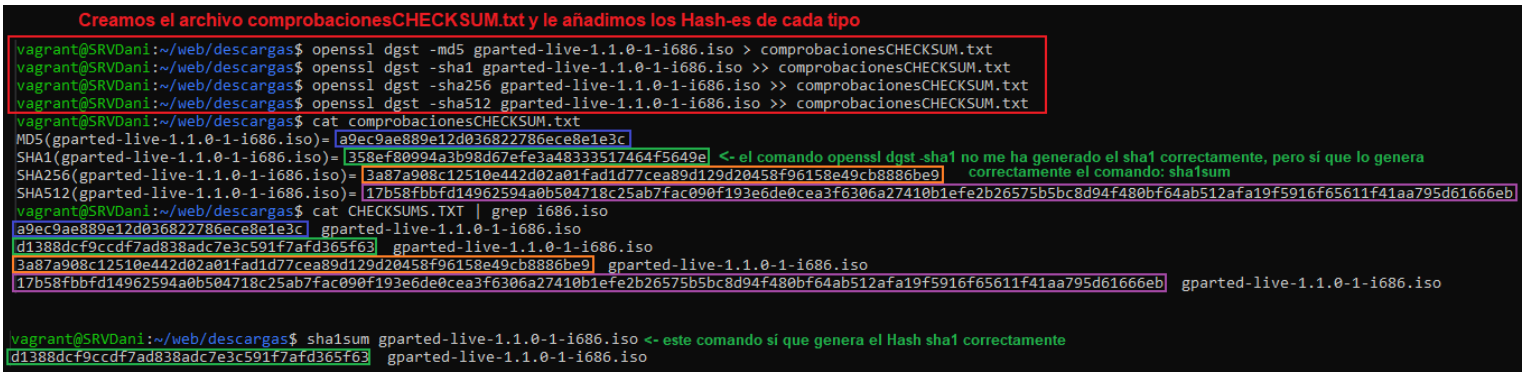
- Para comprobar el hash existen comandos específicos de bash: *$ sha1sum, $ sha256sum, sha512sum, md5sum* … pero usaremos (para empezar a conocerla) la suite criptográfica **openssl**
- Comprueba los diferentes tipos de hash con *$openssl dgst –shaxx miFichero.iso* ¿Coinciden con los publicados en la web?

Antes de nada, voy a dejar plasmado lo siguiente, y es que, utilizando el comando: **$ wget enlace** descargábamos un archivo que, al generarle el hash, nos generaba un hash diferente al que debería. En esta foto explico lo explico:



(si no ves bien la imagen, haz clic aquí)

Por lo que, como digo en el último párrafo en la imagen (el párrafo morado), hemos descargado el .iso que ha subido el profesor a la página del centro, y he realizado el mismo proceso con dicha imagen. Ahora sí que ha funcionado:



(si no ves bien la imagen, haz clic aquí)

Como curiosidad, el comando **$ openssl dgst -sha1 archivo.iso** no me ha generado correctamente el Hash (aunque al profesor y a mis compañeros sí), pero el comando **$ sha1sum archivo.iso** sí que lo ha generado correctamente.

Vamos ahora a hacer una pequeña prueba con un fichero de texto:

- Crea un fichero "declaracionXX.txt" con el siguiente texto: "Me comprometo a dar 10 euros mensuales a cada uno de mis compañeros". Obtén el hash md5. Cambia el texto y coloca 100 en lugar de 10. Vuelve a generar el hash, ¿coinciden?

```
vagrant@SRVDani:~/web$ echo "Me comprometo a dar 10 euros mensuales a cada uno de mis compañeros, menos a Ander Zugasti." > declaracion10.txt
vagrant@SRVDani:~/web$ openssl dgst -md5 declaracion10.txt
MD5(declaracion10.txt)= 01c6c4bec9e2cce184f5f4f9ef530e46
vagrant@SRVDani:~/web$ echo "Me comprometo a dar 100 euros mensuales a cada uno de mis compañeros, menos a Ander Zugasti." > declaracion10.txt
vagrant@SRVDani:~/web$ openssl dgst -md5 declaracion10.txt
MD5(declaracion10.txt)= 0d790a50fc9e2466f9b6ba3e552652ce
vagrant@SRVDani:~/web$ y bualá! distinto!
```

El hash, por tanto, se usa para garantizar la **INTEGRIDAD** de la información: pero todo esto está muy ligado a saber quién nos ha enviado el mensaje: saber de quién es, y que no se ha modificado por el camino … aquí llega de nuevo la firma digital, pero ahora en combinación con el hash …

Hashing y firma digital

Importante: he reestructurado un poco el cómo estaba ordenado el enunciado, ya que en cierto punto te pide que cambies el contenido del fichero CHECKSUMS.TXT para luego comprobar que la firma no es válida, y luego más tarde te pide que lo verifiques para que sí lo sea, así que primero veré que en efecto la firma es válida, y después modificaré CHECKSUMS.TXT para ver como con un simple cambio minúsculo la firma ya no es válida.

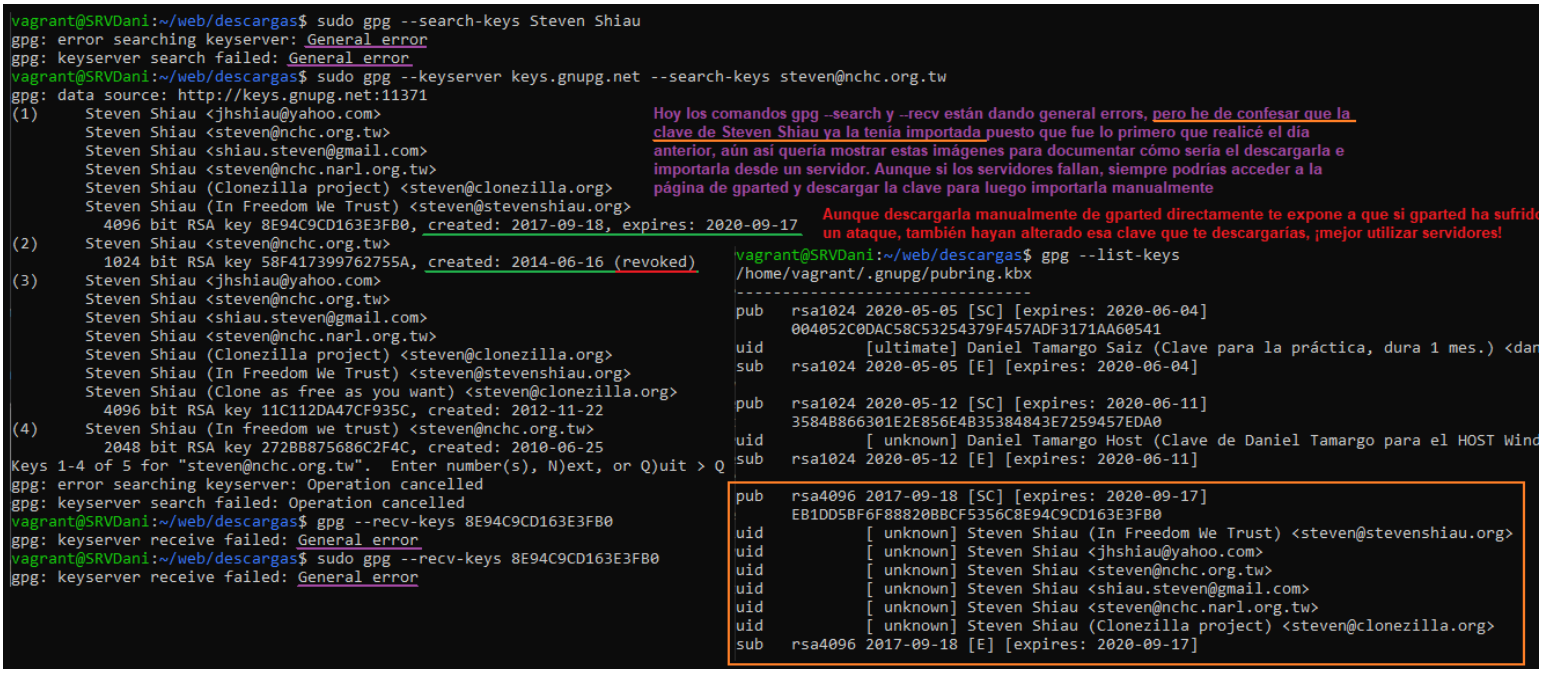
También lo reordeno para que el proceso sea: descargar clave, verificar, alterar, verificar de nuevo, etc...

Lee este artículo antes de empezar.

Has descargado una ISO y comprobado su HASH. ¿Estás seguro de que nadie te ha hecho un ManInTheMiddle y te ha colado si ISO y/o ha hackeado la página de donde la has descargado para colocar ahí su propio fichero de HASH? ¿Cómo sabes que la ISO y el HASH proceden de GPARTED? Necesitas que alguien de GPARTED firme que eso es así y tú puedas verificar esa firma.

- Ya tienes el HASH de la ISO descargado del ejercicio anterior. Descarga ahora el fichero GPG con la firma de ese hash por parte del desarrollador de GPARTED "GParted live maintainer Steven Shiau".
  - Ahora se trataría de verificar que la firma es correcta: procederás a desencriptar la firma del HASH (fichero GPG) con la clave pública de Steven Shiau y el resultado debe ser igual al HASH original:
    - $K_{publicaStevenShiau}$(fichero.gpg firmado con $K_{privadaStevenShiau}$) = fichero hash → Si se desencripta con la clav pública de Steven Shiau, solo Steven Shiau ha podido crearlo, porque es el único que tiene su clave privada!.
    - ¡¡Estaría asegurándome que el hash que yo tengo lo ha creado el desarrollador del GPARTED y no un atacante!!

- Busca la clave pública del desarrollador de GPARTED en alguno de los enlaces que se proporcionan. Lo típico es descargar la clave pública de un servidor diferente, de forma que si alguien está haciendo un MiM y trata de colarte su ISO junto con el hash asociado o ha hackeado la web y colocado su hash, debería también conseguir suplantar al segundo servidor para colarte su clave pública cuando tratas de descargarla.



(ves mal la imagen? haz clic aquí para verla en la nube)

- Para hacerlo, usa el comando *$ gpg –verify ficheroFirma ficheroHash* ¿Te encuentras algún problema?, por qué pide la clave pública y no la privada? ¿de quién es esa clave pública?
  - NOTA: si el fichero de firma hubiera contenido el mensaje original, bastaría con haber hecho *$ gpg –verify ficheroFirma*
  - ¿Por qué nos da un warning? ¿Tiene algo que ver con lo visto en el ejercicio anterior de firma de mensajes?

No tengo problemas puesto que he reorganizado el fichero para que el enunciado tenga más sentido. Pero el problema en cuestión, sería no tener la clave pública de Steven Shiau, puesto que necesitamos esa clave para desencriptar o mejor dicho, verificar el archivo CHECKSUMS.TXT utilizando el fichero firmado CHECKSUMS.TXT.gpg. Utiliza la clave privada para firmarlo, para así distribuir la clave pública (la privada nunca debe distribuirse) y así los demás usuarios, como yo por ejemplo, podremos verificar la firma.

Se dice que, cuando se usa esta forma para firmar un archivo y darle integridad al otro, es una firma detached.

Nos da un warning diciendo que esa clave pública no está verificada (avalada). Eso es bastante normal porque la hemos obtenido por medio de una descarga directa desde un servidor externo. Esto nos lleva al siguiente problema criptográfico, que es el de la **distribución segura de claves públicas**. Hoy día esto se hace mediante certificados firmados por entidades de confianza.

En la web es algo muy común. Los servidores web se configuran para que sirvan páginas con HTTPS (S=Secure), donde toda la comunicación con el cliente web va encriptada. Para poder llevar a cabo el proceso criptográfico, el servidor debe hacer llegar a los clientes su clave pública. Es la PKI = Public Key Infraestructure, y se basa en emisión de certificados firmados por CA = Certification Authorities, que son entidades de confianza. Es el modelo usado por HTTPS en la web.

Para verificarlo, hay dos formas de hacerlo (la primera parece lo más óptimo):

- 1: en una línea especificarle la firma y directamente cuál es su archivo enlazado:

```
vagrant@SRVDani:~/web/descargas$ gpg --verify CHECKSUMS.TXT.gpg CHECKSUMS.TXT
gpg: Signature made Tue 21 Jan 2020 03:46:28 AM UTC
gpg:                using RSA key EB1DD5BF6F88820BBCF5356C8E94C9CD163E3FB0
gpg: Good signature from "Steven Shiau (In Freedom We Trust) <steven@stevenshiau.org>" [unknown]
gpg:                 aka "Steven Shiau <jhshiau@yahoo.com>" [unknown]
gpg:                 aka "Steven Shiau <steven@nchc.org.tw>" [unknown]
gpg:                 aka "Steven Shiau <shiau.steven@gmail.com>" [unknown]
gpg:                 aka "Steven Shiau <steven@nchc.narl.org.tw>" [unknown]
gpg:                 aka "Steven Shiau (Clonezilla project) <steven@clonezilla.org>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: EB1D D5BF 6F88 820B BCF5  356C 8E94 C9CD 163E 3FB0
```

- 2: pedirle que lo desencripte para comprobarlo con el archivo .txt

```
vagrant@SRVDani:~/web/descargas$ gpg --decrypt CHECKSUMS.TXT.gpg
Detached signature.
Please enter name of data file: CHECKSUMS.TXT
gpg: Signature made Tue 21 Jan 2020 03:46:28 AM UTC
gpg:                using RSA key EB1DD5BF6F88820BBCF5356C8E94C9CD163E3FB0
gpg: Good signature from "Steven Shiau (In Freedom We Trust) <steven@stevenshiau.org>" [unknown]
gpg:                 aka "Steven Shiau <jhshiau@yahoo.com>" [unknown]
gpg:                 aka "Steven Shiau <steven@nchc.org.tw>" [unknown]
gpg:                 aka "Steven Shiau <shiau.steven@gmail.com>" [unknown]
gpg:                 aka "Steven Shiau <steven@nchc.narl.org.tw>" [unknown]
gpg:                 aka "Steven Shiau (Clonezilla project) <steven@clonezilla.org>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: EB1D D5BF 6F88 820B BCF5  356C 8E94 C9CD 163E 3FB0
```

  - Edita el hash (fichero TXT) y la firma (fichero TXT.GPG) y comprueba que el fichero de firma no contiene el fichero original … (se dice que la firma está detached)

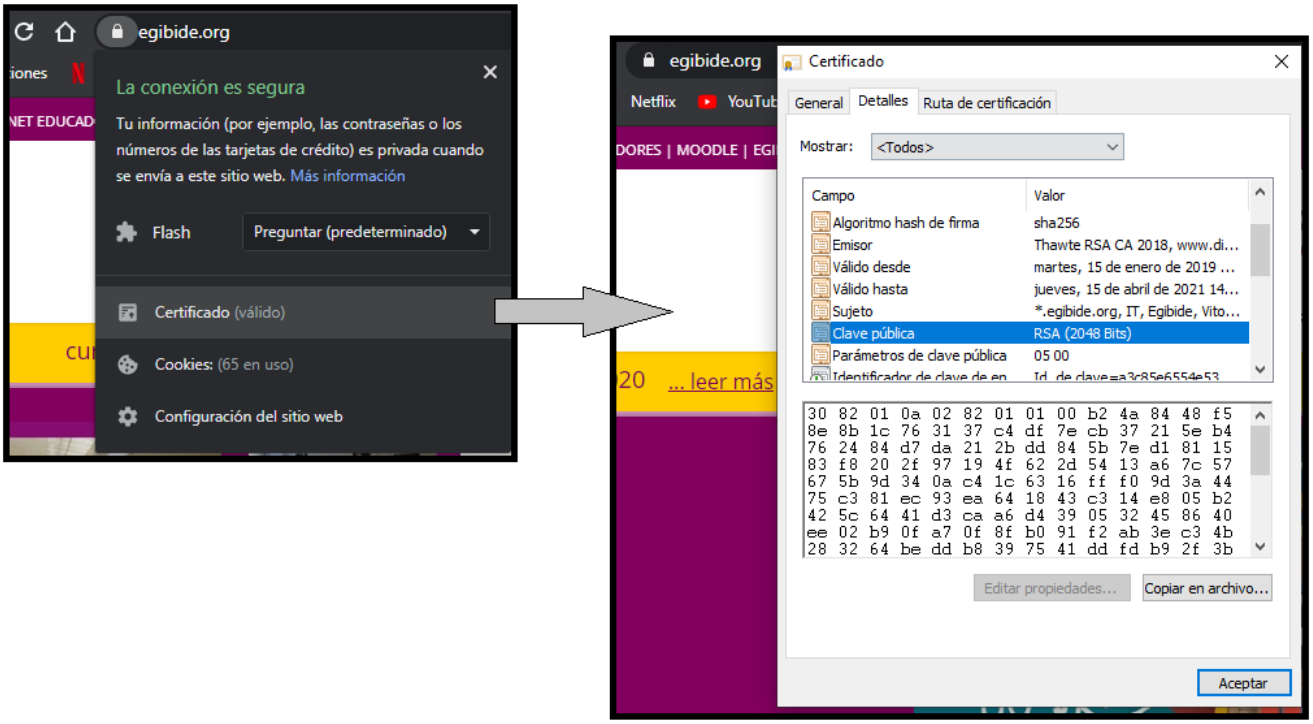
```
vagrant@SRVDani:~/web/descargas$ sudo nano CHECKSUMS.TXT   <- He añadido una simple '#' al principio del fichero
vagrant@SRVDani:~/web/descargas$ gpg --verify CHECKSUMS.TXT.gpg CHECKSUMS.TXT
gpg: Signature made Tue 21 Jan 2020 03:46:28 AM UTC
gpg:                using RSA key EB1DD5BF6F88820BBCF5356C8E94C9CD163E3FB0
gpg: BAD signature from "Steven Shiau (In Freedom We Trust) <steven@stevenshiau.org>" [unknown]

                La firma ya no es válida! Funciona bien!
```

Detached significa separado o suelto, por lo que tiene sentido. Es un sistema ideal para poder acceder de una forma cómoda al fichero CHECKSUMS.TXT, y en caso de no fiarte de un posible ManInTheMiddle o cualquier otra alteración maligna sobre el archivo, puedes utilizar el fichero firmado aunque separado (detached) para verificar el propio CHECKSUMS.TXT.

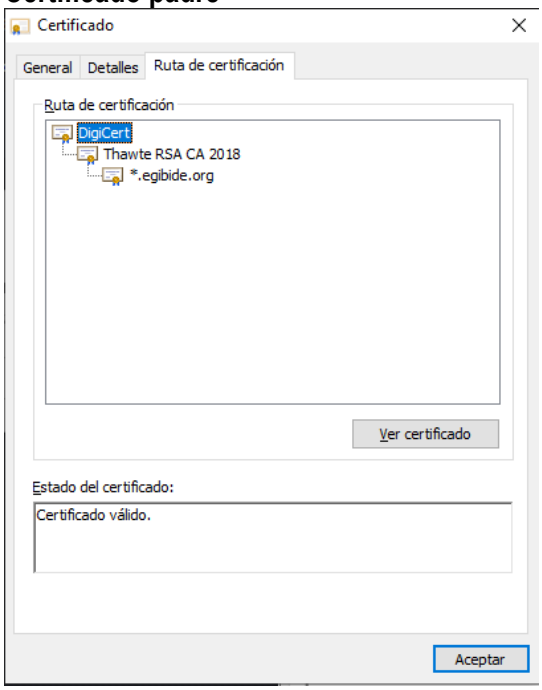
- Para distribuir la clave pública, se utiliza un certificado, que asocia el nombre del sitio web con la clave pública.
    - Conéctate a https://www.egibide.org y consulta su cetificado.
    - Localiza la clave pública
    - ¿A qué nombre está emitido ese certificado? ¿cuál es la fecha de expiración?

Consultar el certificado y localizar y ver la clave pública:

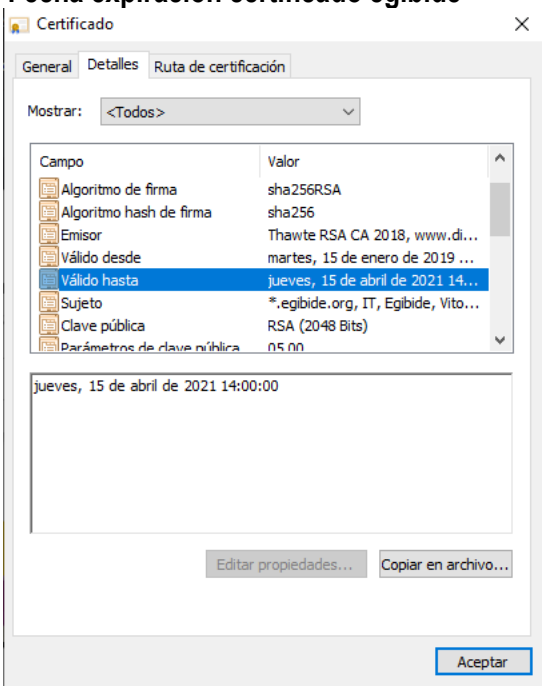


Para ver el certificado del cual "cuelga" o "surge" el certificado de egibide (*.egibide.org), podremos acceder a la pestaña Ruta de certificación. El certificado "padre" o el inicio de la raíz de certificados a la que pertenece el certificado de Egibide es **DigiCert.** Por otro lado, **el certificado de egibide expirará el 15/04/2021.**

Certificado padre



Fecha expiración certificado egibide



- Pero de la misma forma que yo podría hacerme un DNI falso con mi foto diciendo que soy yo, cualquier servidor podría hacerse él mismo un certificado con su clave pública. Hace falta que todo este proceso sea confiable. Chrome debe estar seguro que la clave pública que le ofrece un servidor es suya. Para ello, los certificados deben ser firmados por entidades de confianza CA (Certification Authorities). Puede haber una cadena de varias CAs que terminan en una CA raíz, que viene ya instalada con el propio navegador (Chrome, Edge, Firefox, etc)
    - ¿Cuál es la cadena de confianza del certificado de www.egibide.org ?

Repetido. Lo acabo de responder, ¡esto ya es un abuso!

$ openssl dgst -sha1 ejerciciorepetido.iso
SHA1(ejerciciorepetido.iso)= n0pu3d3sh4c3rm33st0p0rf4v0r

Quizás cuando acabe los proyectos le de caña a estos puntos optativos solo por curiosear un poco.

Optativo: comprobar la ISO de ArchLinux

Repite el proceso en el caso de esta otra distribución.

Optativo: dejar disponible en tu Apache un fichero, su hash y la firma

Como repaso de todo lo anterior, simula que dejas a disposición de los usuarios en tu servidor Apache una imagen para su descarga, junto con su hash y una firma del hash. La imagen se dejará en el directorio /var/www/html/public/img. Utiliza la suite gpg

Sigue los siguientes pasos:

- Crea el hash de la imagen que quieras dejar disponible para su descarga con md5, sha1, sha256 y sha512 en el mismo fichero. Usa recursivamente con > y >> *$openssl dgst –sha256 imagen.jpg > imageXX.jpg.hash*
- Firma el hash con gpg dejando el hash en texto plano y con solo la firma (no incluyendo el mensaje original) *$ gpg --detach-sig --armor imageXX.jpg.hash.* Verifica que en el fichero de firma solo está la firma y no el hash original
    - En realidad hay otros métodos para firmar: --sign == fichero comprimido + firma –clearsign == fichero original en plano + firma en plano, y el que usamos ahora –detach-sig == solo firma, con la opción –armor para que aparezca en ASCII https://access.redhat.com/solutions/1541303
- Deja la imagen, su hash y la firma del hash en /var/www/html/public/img
- Descarga la imagen desde la máquina Windows y haz el proceso de verificación del hash y la firma
    - Verifica el hash con PowerShell
        - `Get-FileHash fichero -Algorithm SHA384`
    - Verifica la firma con https://www.gpg4win.org