MIGUEL CÓRDOBA ARANDA 2ºASIR SEGURIDAD

INTEGRIDAD, FIRMAS Y AUTENTICACIÓN

TAREA 1. Firmas electrónicas (3 puntos)

1. Manda un documento y la firma electrónica del mismo a un compañero. Verifica la firma que tu has recibido.

Para verificar la firma que me ha enviado el compañero **Omar** he usado el siguiente comando:

#gpg --verify documento_firmado.txt.sig.

```
root@MCA:/home/miguel/Descargas# gpg --verify documento_firmado.txt.sig
gpg: asumiendo que los datos firmados están en 'documento_firmado.txt'
gpg: Firmado el lun 15 nov 2021 08:51:34 CET
gpg:                usando RSA clave E152B3B11133E7BB6C1054B7CA261D60644AC899
gpg: Firma correcta de "omar elhani <omar.elhani1@gmail.com>" [desconocido]
gpg: ATENCIÓN: ¡Esta clave no está certificada por una firma de confianza!
gpg:          No hay indicios de que la firma pertenezca al propietario.
Huellas dactilares de la clave primaria: E152 B3B1 1133 E7BB 6C10  54B7 CA26 1D60 644A
C899
```

2. ¿Qué significa el mensaje que aparece en el momento de verificar la firma?

```
 gpg: Firma correcta de "Pepe D <josedom24@gmail.com>" [desconocido]
 gpg: ATENCIÓN: ¡Esta clave no está certificada por una firma de
confianza!
 gpg:          No hay indicios de que la firma pertenezca al propietario.
 Huellas dactilares de la clave primaria: E8DD 5DA9 3B88 F08A DA1D  26BF
5141 3DDB 0C99 55FC
```

Significa que la firma está correcta, pero no hay nada que te asegure que pertenezca la firma a dicha persona por que no está certificada.

3. Vamos a crear un anillo de confianza entre los miembros de nuestra clase, para ello.

3.1 Tu clave pública debe estar en un servidor de claves.

Para este paso he usado el comando: **#gpg --keyserver pgp.rediris.es --send-key <UUID>**.

3.3 Te debes bajar al menos tres claves públicas de compañeros. Firma estas claves.

Las claves que me he bajado son la de tres compañeros: Omar, Antonio y Adrián.

Para ello he usado el comando **#gpg --keyserver pgp.rediris.es --recv-key <UID rediris>**.

```
miguel@MCA:~$ gpg --keyserver pgp.rediris.es --recv-key 644AC899
gpg: clave CA261D60644AC899: clave pública "omar elhani <omar.elhani1@gmail.com>" importada
gpg: Cantidad total procesada: 1
gpg:               importadas: 1
miguel@MCA:~$ gpg --keyserver pgp.rediris.es --recv-key 2CC0D40C
gpg: clave C42384152CC0D40C: clave pública "Antonio Castro Díaz <antoniocastro-@hotmail.com>" importada
gpg: Cantidad total procesada: 1
gpg:               importadas: 1
miguel@MCA:~$ gpg --keyserver pgp.rediris.es --recv-key C39B578A
gpg: clave 6B4E1B2DC39B578A: clave pública "Adrian Diaz <adriandiazaguilar@gmail.com>" importada
gpg: Cantidad total procesada: 1
gpg:               importadas: 1
```

Para firmar las claves he usado **#gpg --sign-key <UID rediris>** .

```
pub   rsa3072 2021-11-11 [SC] [caduca: 2023-11-11]
      E152B3B11133E7BB6C1054B7CA261D60644AC899
uid         [   total   ] omar elhani <omar.elhani1@gmail.com>
sig 3        CA261D60644AC899 2021-11-11  omar elhani <omar.elhani1@gmail.com>
sig          8E927A64B98AC74C 2021-11-18  Miguel Córdoba (prueba) <miguelcor.rrss@gmail.com>
sub   rsa3072 2021-11-11 [E] [caduca: 2023-11-11]
sig          CA261D60644AC899 2021-11-11  omar elhani <omar.elhani1@gmail.com>

pub   rsa3072 2021-11-11 [SC] [caduca: 2023-11-11]
      7265A1244839F6D3FC563792C42384152CC0D40C
uid         [   total   ] Antonio Castro Díaz <antoniocastro-@hotmail.com>
sig 3        C42384152CC0D40C 2021-11-11  Antonio Castro Díaz <antoniocastro-@hotmail.com>
sig          8E927A64B98AC74C 2021-11-18  Miguel Córdoba (prueba) <miguelcor.rrss@gmail.com>
sub   rsa3072 2021-11-11 [E] [caduca: 2023-11-11]
sig          C42384152CC0D40C 2021-11-11  Antonio Castro Díaz <antoniocastro-@hotmail.com>

pub   rsa3072 2021-11-12 [SC] [caduca: 2023-11-12]
      E26E125851F84B4A3BC066316B4E1B2DC39B578A
uid         [   total   ] Adrian Diaz <adriandiazaguilar@gmail.com>
sig 3        6B4E1B2DC39B578A 2021-11-12  Adrian Diaz <adriandiazaguilar@gmail.com>
sig          8E927A64B98AC74C 2021-11-18  Miguel Córdoba (prueba) <miguelcor.rrss@gmail.com>
sub   rsa3072 2021-11-12 [E] [caduca: 2023-11-12]
sig          6B4E1B2DC39B578A 2021-11-12  Adrian Diaz <adriandiazaguilar@gmail.com>
```

3.4 Tu te debes asegurar que tu clave pública es firmada por al menos tres compañeros de la clase.

Para ver la cantidad de veces que me han firmado mi clave he usado el comando: **#gpg --list-keys**.

```
root@MCA:/home/miguel/Descargas# gpg --import clave-miguel.asc
gpg: clave 93E00F9A8C74FBC0: "Miguel Cordoba <miguelcor.rrss@gmail.com>" 1 firma nueva
gpg: Cantidad total procesada: 1
gpg:         nuevas firmas: 1
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: nivel: 0  validez:   1  firmada:   3  confianza: 0-, 0q, 0n, 0m, 0f, 1u
gpg: nivel: 1  validez:   3  firmada:   2  confianza: 3-, 0q, 0n, 0m, 0f, 0u
gpg: siguiente comprobación de base de datos de confianza el: 2021-12-11
```

3.5 Una vez que firmes una clave se la tendrás que devolver a su dueño, para que otra persona se la firme.

Para devolverle la clave a su dueño primero he tenido que exportarla a un fichero y luego pasarle el fichero por correo.

#gpg --export -a 644AC899 > firmado_ omar(OMAR)

#gpg --export -a 2CC0D40C > firmado_antonio (ANTONIO CASTRO)

#gpg --export -a C39B578A > firmado_adrian (ADRIÁN DIAZ)

3.6 Cuando tengas las tres firmas sube la clave al servidor de claves y rellena tus datos en la tabla [Claves públicas PGP 2020-2021] .

Claves públicas PGP 2021-2022

NOMBRE

Raúl Ruiz Padilla	3D608DE0	pgp.rediris.es
Omar Elhani	644AC899	pgp.rediris.es
Alejandro Gutiérrez	2330736A	pgp.rediris.es
Miguel Cordoba	8C74FBC0	pgp.rediris.es

TAREA 2: Correo seguro con evolution/thunderbird (2 puntos)

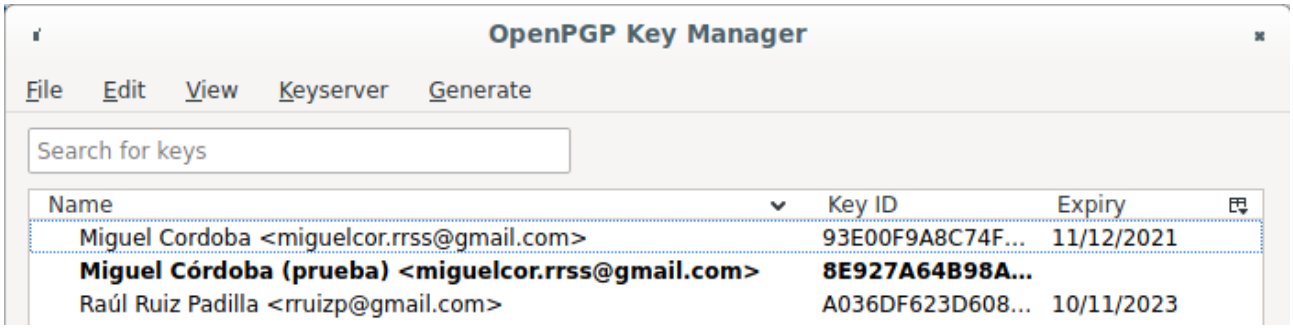
1. Configura el cliente de correo evolution con tu cuenta de correo habitual.

Para configurar **Thunderbird** con mi cuenta **gmail** primero he tenido que ingresar mi correo **gmail** y mi contraseña, después he tenido que confirmar mi cuenta en **gmail**, por último he accedido y se han sincronizado las bandejas de entradas. Posteriormente he configurado mi cliente **Thunderbird** para mandar mensajes cifrados.

2. Añade a la cuenta las opciones de seguridad para poder enviar correos firmados con tu clave privada o cifrar los mensajes para otros destinatarios.

Para realizar este ejercicio primero he exportado mi clave pública, mi clave privada y la clave del destinatario, que en este caso es **Raúl Ruíz Padilla.** Para la exportación he realizado estos pasos.

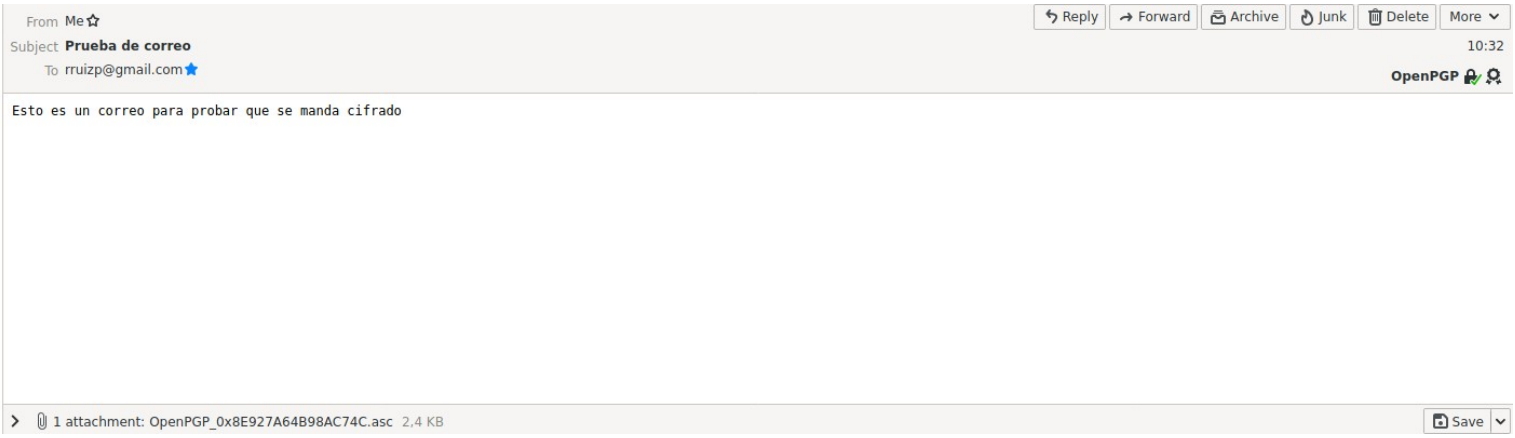
1. He entrado en **Thunderbird > Account Setting > End to End Encryption > OpengPGP > OpenPGP Key Manager**.



En la pestaña **File > Import Public Key from File** he importado mi clave pública y la de Raúl y por último en **File > Import Secret Key from File** he importado mi clave privada.

3. Envía y recibe varios mensajes con tus compañeros y comprueba el funcionamiento adecuado de GPG

En la pestaña **Write > Security** he seleccionado **Require Encryption** y he escrito un texto.



TAREA 3: Integridad de Ficheros (1 punto)

1. Para validar el contenido de la imagen CD, solo asegúrese de usar la herramienta apropiada para sumas de verificación. Para cada versión publicada existen archivos de suma de comprobación con algoritmos fuertes (SHA256 y SHA512); debería usar las herramientas `sha256sum` o `sha512sum` para trabajar con ellos.

Para realizar este ejercicio me he descargado los fichero **SHA256SUMS** y **SHA512SUMS**, los respectivos ficheros firmados **SHA256SUMS.sign** y **SHA512SUMS.sign**, junto con una ISO de instalación de Debian11(**debian-11.1.0-amd64-netinst.iso**). La web para descargarselo es https://cdimage.debian.org/debian-cd/current/amd64/iso-cd/ .

Name	Last modified	Size
Parent Directory		-
SHA256SUMS	2021-10-09 22:45	302
SHA256SUMS.sign	2021-10-09 22:53	833
SHA512SUMS	2021-10-09 22:45	494
SHA512SUMS.sign	2021-10-09 22:53	833
debian-11.1.0-amd64-netinst.iso	2021-10-09 14:30	378M
debian-edu-11.1.0-amd64-netinst.iso	2021-10-09 14:30	439M
debian-mac-11.1.0-amd64-netinst.iso	2021-10-09 14:30	377M

El siguiente paso ha sido comprobar la suma de verificación pasando los ficheros **SHA256SUMS** y y **SHA512SUMS** a la ISO. Para ello he usado los comandos:

#sha256sum -c SHA256SUMS 2>/dev/null | egrep “debian-11.1.0-amd64-netinst.iso”.

#sha512sum -c SHA512SUMS 2>/dev/null | egrep “debian-11.1.0-amd64-netinst.iso”.

```
miguel@MCA:~/Descargas$ sha256sum -c SHA256SUMS 2> /dev/null | egrep "debian-11.1.0-amd64-netinst.iso"
debian-11.1.0-amd64-netinst.iso: La suma coincide
miguel@MCA:~/Descargas$ sha512sum -c SHA512SUMS 2> /dev/null | egrep "debian-11.1.0-amd64-netinst.iso"
debian-11.1.0-amd64-netinst.iso: La suma coincide
```

NOTA: Importante poner las comillas en el nombre de la ISO porque de lo contrario te muestra una salida erronea.

2. Verifica que el contenido del hash que has utilizado no ha sido manipulado, usando la firma digital que encontrarás en el repositorio. Puedes encontrar una guía para realizarlo en este artículo: [How to verify an authenticity of downloaded Debian ISO images](#).

Posteriormente he verificado los ficheros firmados y me han indicado que la firma de Debian es **INCORRECTA**. Para la verificación de dichos fichero he ejecutado:

#gpg --verify SHA256SUMS.sign SHA256SUMS.

#gpg --verify SHA512SUMS.sign SHA512SUMS.

```
miguel@MCA:~/Descargas$ gpg --verify SHA256SUMS.sign SHA256SUMS
gpg: Firmado el sáb 09 oct 2021 22:53:47 CEST
gpg:                usando RSA clave DF9B9C49EAA9298432589D76DA87E80D6294BE9B
gpg: Firma INCORRECTA de "Debian CD signing key <debian-cd@lists.debian.org>" [desconocido]
miguel@MCA:~/Descargas$ gpg --verify SHA512SUMS.sign SHA512SUMS
gpg: Firmado el sáb 09 oct 2021 22:53:48 CEST
gpg:                usando RSA clave DF9B9C49EAA9298432589D76DA87E80D6294BE9B
gpg: Firma INCORRECTA de "Debian CD signing key <debian-cd@lists.debian.org>" [desconocido]
```

TAREA 4: INTEGRIDAD Y AUTENTICIDAD (APT SECURE)

1. **¿Qué software utiliza apt secure para realizar la criptografía asimétrica?.**
   La herramienta que se **utiliza** en **Secure-Apt** para firmar los ficheros y comprobar sus firmas es **GPG** (GNU Privacy Guard).

2. **¿Para que sirve el comando `apt-key`? ¿Qué muestra el comando `apt-key list`?** .
   Apt-key → Herramienta usada para gestionar el anillo de llaves de **gpg** para asegurar **Apt**.

```
root@MCA:/home/miguel# apt-key
Usage: apt-key [--keyring file] [command] [arguments]

Manage apt's list of trusted keys

  apt-key add <file>          - add the key contained in <file> ('-' for stdin)
  apt-key del <keyid>         - remove the key <keyid>
  apt-key export <keyid>      - output the key <keyid>
  apt-key exportall           - output all trusted keys
  apt-key update              - update keys using the keyring package
  apt-key net-update          - update keys using the network
  apt-key list                - list keys
  apt-key finger              - list fingerprints
  apt-key adv                 - pass advanced options to gpg (download key)

If no specific keyring file is given the command applies to all keyring files.
```

Apt-key list → Lista las claves públicas de los repositorios que tienes en tu máquina.

```
pub   rsa2048 2015-10-28 [SC]
      BC52 8686 B50D 79E3 39D3  721C EB3E 94AD BE12 29CF
uid        [desconocida] Microsoft (Release signing) <gpgsecurity@microsoft.com>

pub   rsa4096 2020-05-07 [SC]
      E8A0 32E0 94D8 EB4E A189  D270 DA41 8C88 A321 9F7B
uid        [desconocida] HashiCorp Security (HashiCorp Package Signing) <security+packaging@hashicorp.com>
sub   rsa4096 2020-05-07 [E]

pub   dsa1024 2010-05-18 [SC]
      7B0F AB3A 13B9 0743 5925  D9C9 5442 2A4B 98AB 5139
uid        [desconocida] Oracle Corporation (VirtualBox archive signing key) <info@virtualbox.org>
sub   elg2048 2010-05-18 [E]

/etc/apt/trusted.gpg.d/debian-archive-bullseye-automatic.gpg
------------------------------------------------------------
pub   rsa4096 2021-01-17 [SC] [caduca: 2029-01-15]
      1F89 983E 0081 FDE0 18F3  CC96 73A4 F27B 8DD4 7936
uid        [desconocida] Debian Archive Automatic Signing Key (11/bullseye) <ftpmaster@debian.org>
sub   rsa4096 2021-01-17 [S] [caduca: 2029-01-15]

/etc/apt/trusted.gpg.d/debian-archive-bullseye-security-automatic.gpg
---------------------------------------------------------------------
pub   rsa4096 2021-01-17 [SC] [caduca: 2029-01-15]
      AC53 0D52 0F2F 3269 F5E9  8313 A484 4904 4AAD 5C5D
uid        [desconocida] Debian Security Archive Automatic Signing Key (11/bullseye) <ftpmaster@debian.org>
sub   rsa4096 2021-01-17 [S] [caduca: 2029-01-15]
```

3. **¿En que fichero se guarda el anillo de claves que guarda la herramienta `apt-key`? .**
   El directorio donde se guardan las claves es **/etc/apt/trusted.gpg.d**

```
root@MCA:/home/miguel# ls /etc/apt/trusted.gpg.d
ansible_ubuntu_ansible.gpg
debian-archive-bullseye-automatic.gpg
debian-archive-bullseye-security-automatic.gpg
debian-archive-bullseye-stable.gpg
debian-archive-buster-automatic.gpg
debian-archive-buster-security-automatic.gpg
debian-archive-buster-stable.gpg
debian-archive-stretch-automatic.gpg
debian-archive-stretch-security-automatic.gpg
debian-archive-stretch-stable.gpg
gezakovacs_ubuntu_ppa.gpg
google-chrome.gpg
microsoft.gpg
```

   y el fichero es **trusted.gpg** que se encuentra en **/etc/apt**.

4. **¿Qué contiene el archivo `Release` de un repositorio de paquetes?. ¿Y el archivo `Release.gpg`?. Puedes ver estos archivos en el repositorio `http://ftp.debian.org/debian/dists/Debian10.1/`. Estos archivos se descargan cuando hacemos un `apt update`.**

   Release → Contiene los hash **MD5** y **SHA256** de los ficheros **Packages.*gz*** y **Sources.*gz*.** Esto permite a **APT** comprobar que el fichero **Packages.*gz*** y/o el fichero **Sources**.*gz* que se haya descargado es el original y no ha sufrido ningún cambio fortuito o intencionado.

   Release.gpg → Es la firma digital del fichero **Release**, con lo cual se garantiza la seguridad de dicho fichero. Es en este punto donde empieza la cadena de comprobación de la seguridad de cualquier fichero que nos bajemos de los repositorios de Debian.

5. **Explica el proceso por el cual el sistema nos asegura que los ficheros que estamos descargando son legítimos.**

   Cuando llevamos a cabo un **apt update**, se descargan los ficheros **Packages.gz**, **Release** y **Release.gpg**, ficheros que son necesarios para la hora de la descarga del paquete, pues lo primero que se hace es comprobar que el *hash* del paquete **.deb** coincide con el indicado en el fichero **Packages.gz**, comprobando a su vez que coincide con el indicado en el fichero **Release**, comprobando por último la firma de dicho fichero, que se encuentra contenida en el fichero **Release.gpg**.
   Si nos fijamos, se establece una estructura de firmado jerárquica, de manera que ante cualquier mínima manipulación, nos daríamos cuenta, pues el hecho de modificar un único bit haría que el *hash* resultante fuese totalmente distinto.

6. **Añade de forma correcta el repositorio de virtualbox añadiendo la clave pública de virtualbox como se indica en la documentación.**

   Primero añadiremos el repositorio a nuestro **source.list**:

   `#echo "deb https://download.virtualbox.org/virtualbox/debian buster contrib" >> /etc/apt/sources.list`.

   El siguiente paso será añadir su clave pública, porque de lo contrario, el **Release** no podrá verificar la firma y no nos dejará descargar **Virtualbox** del repositorio.

   `#wget -q https://www.virtualbox.org/download/oracle_vbox_2016.asc -O- | sudo apt-key add -.`

   `#wget -q https://www.virtualbox.org/download/oracle_vbox.asc -O- | sudo apt-key add -`.

   A continuación actualizaremos los repositorios **`#apt-get update`** Y por último instalaremos **Virtualbox** con el comando **`#apt-get install virtualbox-6.1`**.

TAREA 5: AUTENTIFICACIÓN: ejemplo SSH (2 Puntos)

1. Explica los pasos que se producen entre el cliente y el servidor para que el protocolo cifre la información que se transmite? ¿Para qué se utiliza la criptografía simétrica? ¿Y la asimétrica?

Cuando un cliente intenta conectarse al servidor a través de TCP, el servidor presenta los protocolos de cifrado y las respectivas versiones que soporta. Si el cliente tiene un par similar de protocolo y versión se inicia una **negociación** e intercambian las claves públicas usando **criptografía asimétrica**.

Si es la primera vez que se conecta, nos preguntará que si confiamos en esa clave y si ya nos hemos conectado varias veces con esa misma clave, lo que hará el servidor será comparar la clave del cliente con la que tiene en el **authorized_key**, si esa comparación es positiva nos dejará conectarnos al servidor.

Cuando nos conectamos al servidor, las dos partes usan un **Algoritmo de Intercambio** para crear una clave simétrica y una vez acabado todo lo anterior el cliente debe autenticarse para permitir el acceso.

2. **Explica los dos métodos principales de autentificación: por contraseña y utilizando un par de claves públicas y privadas.**

Contraseña → Se le pide al usuario la contraseña asociada al usuario al que está tratando de conectarse. Éstas credenciales pasan con seguridad a través del túnel cifrado simétricamente, así que no hay ninguna posibilidad de que sean capturadas por un tercero.

Par de Claves → El usuario tiene su clave pública almacenada en el servidor (concretamente en el fichero **~/.ssh/authorized_keys**), de manera que a la hora de realizar la conexión, el usuario se presenta y le indica al servidor que es el dueño de dicha clave. El servidor desconfía, por lo que le hace un “desafío”, consistente en encriptar un mensaje con dicha clave pública, enviándoselo encriptado al usuario que trata de acceder, el cuál debe descifrarlo haciendo uso de la clave privada asociada a dicha clave pública.

3. **En el cliente para que sirve el contenido que se guarda en el fichero `~/.ssh/know_hosts`?**

Para no tener que preguntar si confiamos en la clave pública con la que nos conectamos cada vez que hagamos una conexión ssh, sino que revisa ese fichero y si existe esa clave establecerá la conexión automáticamente.

4. **¿Qué significa este mensaje que aparece la primera vez que nos conectamos a un servidor?**

```
$ ssh debian@172.22.200.74
The authenticity of host '172.22.200.74 (172.22.200.74)' can't be established.
ECDSA key fingerprint is SHA256:7ZoNZPCbQTnDso1meVSNoKszn38ZwUI4i6saebbfL4M.
Are you sure you want to continue connecting (yes/no)?
```

El cliente tiene la posibilidad de decidir si quiere continuar o no con la conexión, dependiendo si confía en el otro extremo (en el servidor). En caso afirmativo, se añadirá el *host key* al fichero situado en **~/.ssh/known_hosts**, de manera que para la siguiente ocasión, no se preguntará al cliente si confía, sino que mirará en dicho fichero para comprobar su previa existencia.

5. **En ocasiones cuando estamos trabajando en el cloud, y reutilizamos una ip flotante nos aparece este mensaje:**

```
 $ ssh debian@172.22.200.74
 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
 @    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
 IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
 Someone could be eavesdropping on you right now (man-in-the-middle attack)!
 It is also possible that a host key has just been changed.
 The fingerprint for the ECDSA key sent by the remote host is
 SHA256:W05RrybmcnJxD3fbwJOgSNNWATkVftsQl7EzfeKJgNc.
 Please contact your system administrator.
 Add correct host key in /home/jose/.ssh/known_hosts to get rid of this message.
 Offending ECDSA key in /home/jose/.ssh/known_hosts:103
   remove with:
   ssh-keygen -f "/home/jose/.ssh/known_hosts" -R "172.22.200.74"
 ECDSA host key for 172.22.200.74 has changed and you have requested strict
checking.
```

6. **¿Qué guardamos y para qué sirve el fichero en el servidor ~/.ssh/authorized_keys?** .

En el authorized_keys guardamos claves publicas que están autorizadas para establecer una conexión remota a dicha maquina y sirve para no tener que confirmar la conexión remota cada vez que la queramos establecer.