



RAI SI Practica 4 2024

Redes y aplicaciones Internet (Universitat Oberta de Catalunya)



Escanea para abrir en Studocu



Redes y Aplicaciones Internet / Sistemas de Internet

Práctica 4. Cómo evitar intrusos o ataques

Esta práctica consta de cuatro partes:

Parte 1. Integridad con SHA.

Parte 2. Cifrado simétrico con GPG.

Parte 3. Cifrado asimétrico con GPG.

Parte 4. Configuración segura de Apache, conexión con cliente y explicación de SSL.

Calificación

La nota irá en función de las partes entregadas:

Parte 1: permite optar a una nota máxima en esta práctica de C-.

Partes 1 y 2: permite optar a una nota máxima en esta práctica de C+.

Partes 1, 2 y 3: permite optar a una nota máxima en esta práctica de B.

Partes 1, 2, 3 y 4: permite optar a una nota máxima en esta práctica de A.

***Nota:** Entrega las respuestas a los ejercicios, las capturas de pantalla, etc. en el registro de evaluación continua del aula. Si necesitas enviar más de un archivo, usa algún programa de compresión para hacer la entrega en un solo fichero comprimido que incluya todos los contenidos.*

Introducción

Durante esta práctica trabajarás los conceptos de **integridad**, **confidencialidad**, **autenticidad** y **no repudio**, conjuntamente con las técnicas de cifrado simétrico y asimétrico, fundamentales en la seguridad informática. Para ello utilizarás distintas herramientas incluidas en Ubuntu, por lo que desarrollarás la práctica sobre un servidor Linux (escoge el que quieras) y un cliente Linux o Windows. Puedes usar una máquina virtual o una instalación en una máquina real.

Para todos los apartados de los ejercicios donde se pide ejecutar algún comando, **adjunta un pantallazo** donde se vea el comando y la respuesta.

Para las capturas de pantalla que se piden, **configura el prompt del sistema operativo de manera que muestre el día en que estás y tu nombre de usuario.**

A lo largo de la práctica se pide que se ejecuten algunos comandos. **Es indispensable incluir en la respuesta el comando utilizado y el significado de cada uno de los parámetros que se utilicen.**

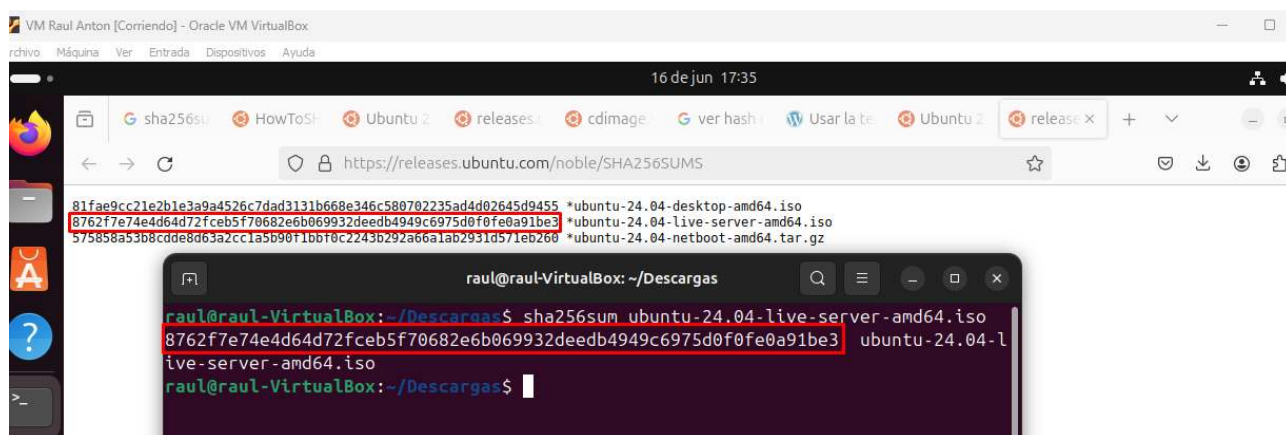
Envía solo tus respuestas a los ejercicios, con los pantallazos correspondientes, al registro de evaluación continua. **No hay que incluir el enunciado de la práctica ni de las preguntas.** Si por algún motivo quieres enviar más de un fichero, usa algún programa de compresión para hacer la entrega en un solo archivo que incluya todos los contenidos.

Primera Parte (nota máxima C-): integridad

Según la Wikipedia, el SHA (*Secure Hash Algorithm*, Algoritmo de Hash Seguro) es una familia de funciones *hash* de cifrado. SHA se utiliza extensamente en el mundo del software para proporcionar la seguridad de que un archivo descargado de Internet no se ha alterado. Comparando el *hash* publicado con la suma de comprobación del archivo descargado, un usuario puede tener la confianza suficiente de que el archivo es igual que el publicado por los desarrolladores. Por lo tanto, es una herramienta que nos permite comprobar la integridad de las descargas, aunque también la podemos usar con archivos que tengamos guardados en nuestro disco duro.

Ejercicio 1

- a) Descarga algún fichero que en la web tenga su SHA disponible. Mediante el comando adecuado en Ubuntu comprueba que el SHA del fichero descargado coincide con el que aparece en la página. Incluye una captura de pantalla donde se vea que coinciden.



- b) Busca información sobre las diferentes funciones que forman la familia SHA. ¿Cuál es la longitud del resumen que produce cada una?

SHA-0 y SHA-1 producen un resumen de 160 bits. SHA-2 tiene varias funciones con diferentes longitudes que van desde los 224 bits hasta los 256.

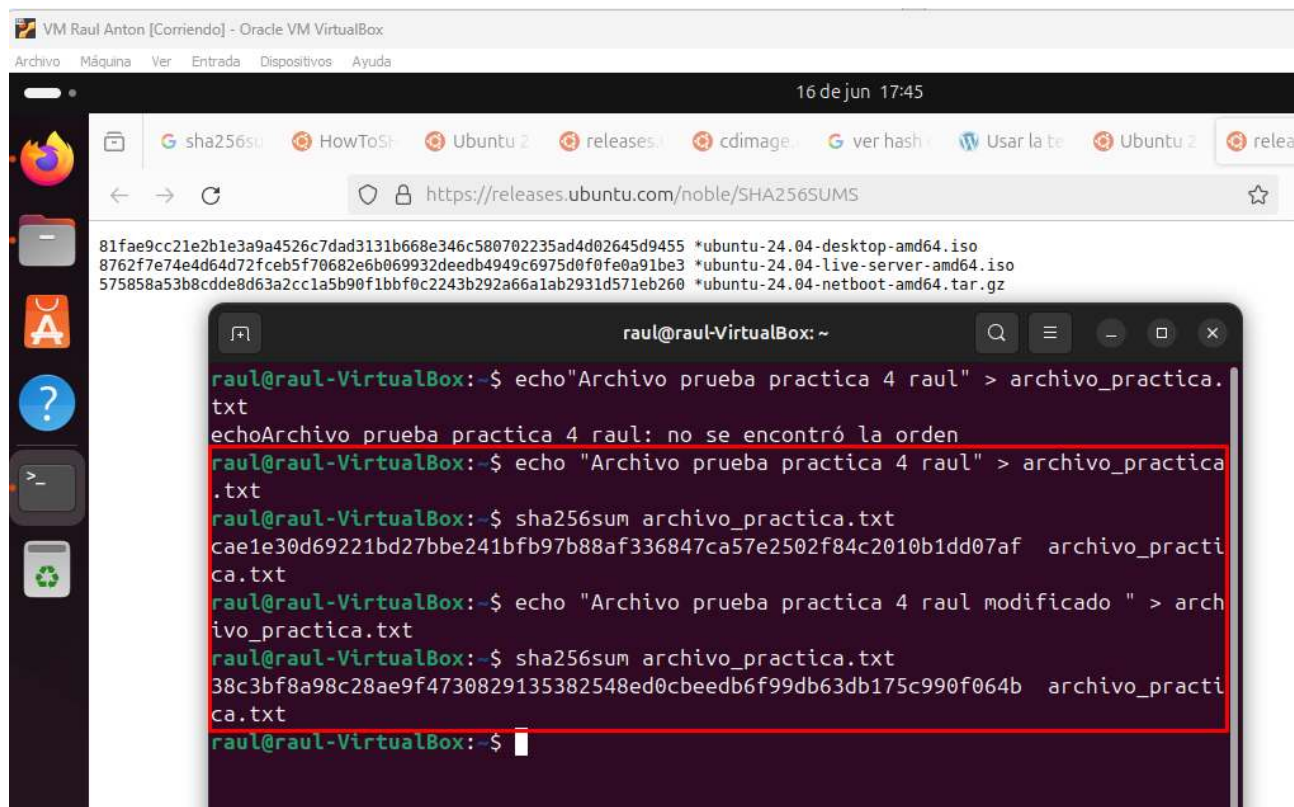
SHA-3 también tiene varias funciones con diferentes longitudes, que van desde los 224 bits hasta los 512.

- c) ¿Qué problema hay asociado al uso de SHA-0? ¿Y SHA-1?

Por un lado, SHA-0 fue retirado debido a que se encontraron vulnerabilidades que permitían ataques de colisión, donde diferentes mensajes producen el mismo hash.

Al igual que el SHA-0, el SHA-1 también ha sido considerado inseguro debido a su vulnerabilidad a ataques de colisión.

- d) Crea un fichero de texto y calcula un *hash* con alguna función de la familia SHA-2. Modifica el fichero y comprueba que el resumen no coincide (captura de pantalla).



The screenshot shows a terminal window titled 'raul@raul-VirtualBox: ~' with the following commands and output:

```
raul@raul-VirtualBox:~$ echo "Archivo prueba practica 4 raul" > archivo_practica.txt
raul@raul-VirtualBox:~$ echo "Archivo prueba practica 4 raul" > archivo_practica.txt
raul@raul-VirtualBox:~$ sha256sum archivo_practica.txt
cae1e30d69221bd27bbe241bfb97b88af336847ca57e2502f84c2010b1dd07af  archivo_practica.txt
raul@raul-VirtualBox:~$ echo "Archivo prueba practica 4 raul modificado " > archivo_practica.txt
raul@raul-VirtualBox:~$ sha256sum archivo_practica.txt
38c3bf8a98c28ae9f4730829135382548ed0cbeedb6f99db63db175c990f064b  archivo_practica.txt
raul@raul-VirtualBox:~$
```

- e) ¿Es posible a partir del *hash* de un texto obtener el texto original?

No es posible obtener el texto original a partir del hash. Las funciones hash están diseñadas para ser unidireccionales, lo que significa que no se puede revertir el hash para obtener el mensaje original

Segunda Parte (nota máxima C+): cifrado simétrico

Pretty Good Privacy (PGP) es un software desarrollado por Phil Zimmermann con el fin de proteger la información compartida a través de Internet y facilitar la autenticación. Gracias a su éxito, la IETF se basó en PGP para definir el estándar OpenPGP (definido en el RFC4880 del IETF), facilitando la correcta comunicación cifrada entre diferentes programas que implementan el mismo protocolo. GNU Privacy Guard (GnuPG) es una implementación libre y completa de OpenPGP. Esta herramienta permite cifrar, así como firmar información y comunicaciones, además de gestionar las claves necesarias para hacerlo.

En esta segunda parte de la práctica utilizaremos la herramienta GnuPG para cifrar y descifrar mensajes (ficheros de texto plano) de forma simétrica. GnuPG suele estar ya instalado en la mayoría de distribuciones Gnu/Linux. Para comprobarlo, utiliza la siguiente orden de consola:

```
gpg --version
```

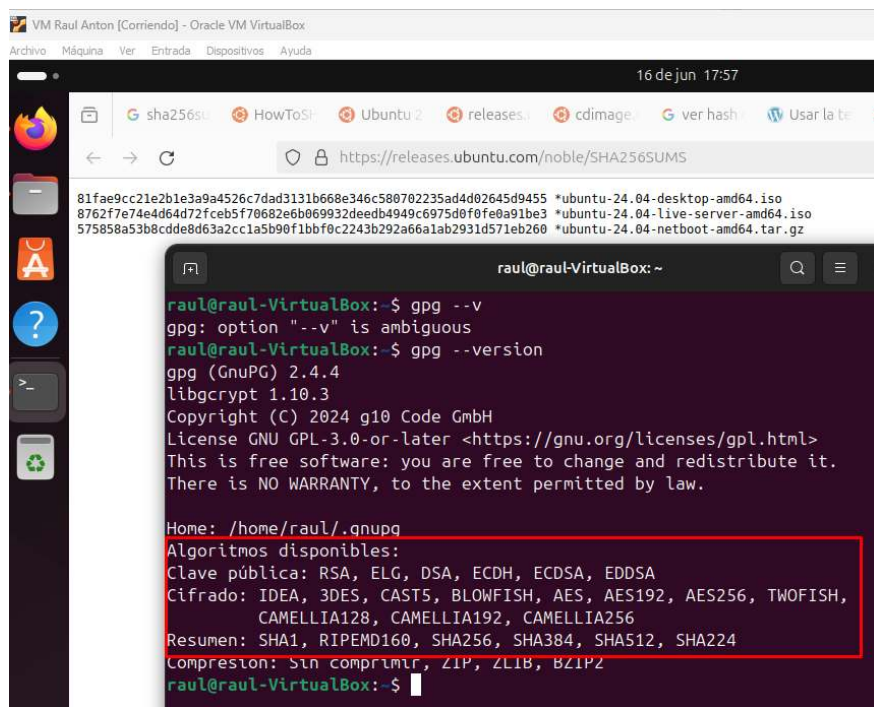
En caso de que el comando no exista instala el software mediante la siguiente orden:

```
sudo apt-get install gnupg
```

Ejercicio 2

- a) En la criptografía simétrica toda la seguridad recae en la clave, por lo que ésta debe ser muy difícil de romper. Cita dos características que hacen que una clave sea difícil de romper.
 - [Cuanto más larga la clave, más difícil será probar todas las combinaciones.](#)
 - [La clave ha de ser aleatoria, sin seguir patrones claros, para dificultar los ataques con análisis de patrones.](#)
- b) Cita tres algoritmos de cifrado simétrico y busca en la red qué longitud de clave tienen.
 - [AES \(Advanced Encryption Standard\): De 128 bits a 256 bits.](#)
 - [DES \(Data Encryption Standard\) : De 56 bits.](#)
 - [3DES: De 112 bits a 168 bits.](#)

- c) Utiliza el comando `gpg` para averiguar qué algoritmos de cifrado simétrico soporta.



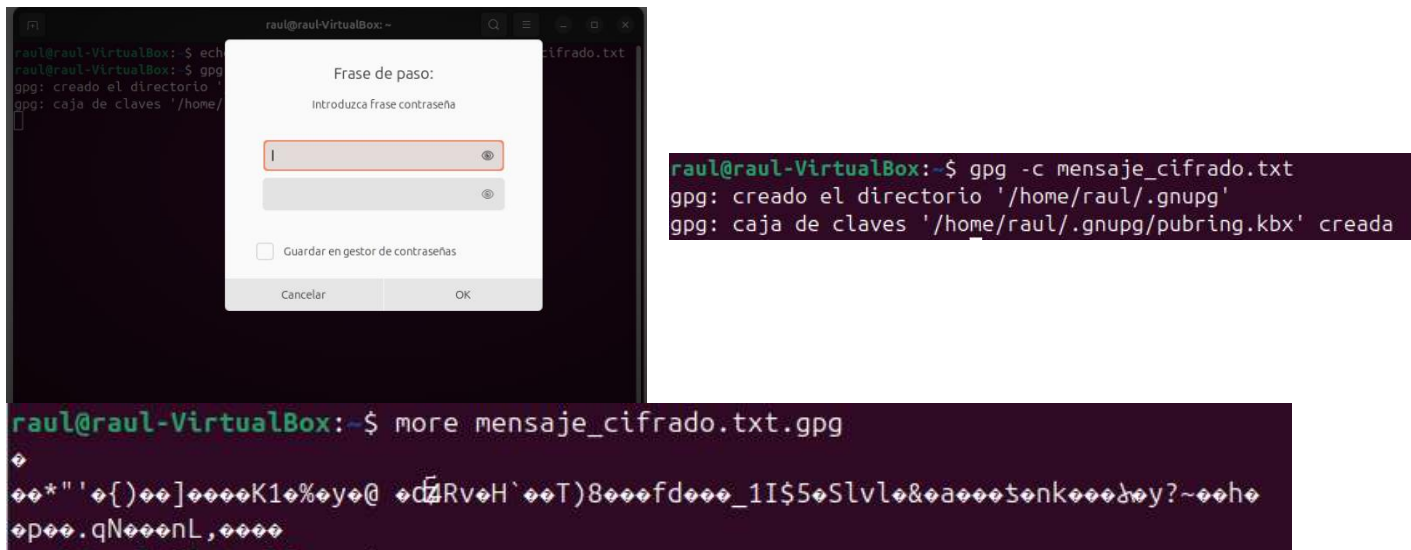
```

raul@raul-VirtualBox: ~
raul@raul-VirtualBox:~$ gpg --v
gpg: option "--v" is ambiguous
raul@raul-VirtualBox:~$ gpg --version
gpg (GnuPG) 2.4.4
libgcrypt 1.10.3
Copyright (C) 2024 g10 Code GmbH
License GNU GPL-3.0-or-later <https://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Home: /home/raul/.gnupg
Algoritmos disponibles:
Clave pública: RSA, ELG, DSA, ECDH, ECDSA, EDDSA
Cifrado: IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH,
CAMELLIA128, CAMELLIA192, CAMELLIA256
Resumen: SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Compresión: Sin comprimir, ZIP, ZLIB, BZIP2
raul@raul-VirtualBox:~$
  
```

Aquí podemos ver los algoritmos disponibles.

- d) Utiliza `man gpg` para indicar con qué opción se puede cifrar de forma simétrica. Crea un fichero de texto con algún contenido y cífralo. Intenta verlo con `cat` o `more`.



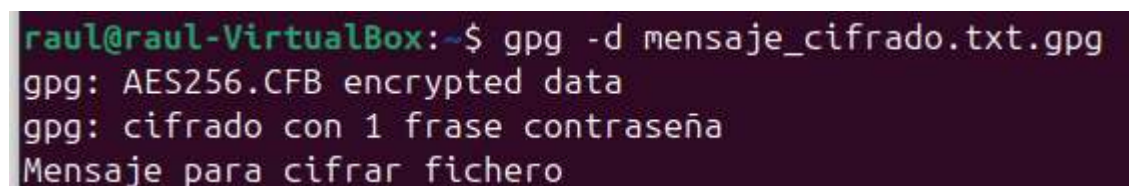
```

raul@raul-VirtualBox:~$ gpg -c mensaje_cifrado.txt
gpg: creado el directorio '/home/raul/.gnupg'
gpg: caja de claves '/home/raul/.gnupg/pubring.kbx' creada

raul@raul-VirtualBox:~$ more mensaje_cifrado.txt.gpg
*~'({}]K1%yo@ dRvH`T)8fddd_1IS5Slvl&aasnkoy?~h
p.qNnL,
  
```

- e) ¿Con qué opción puedes descifrar el fichero? Descifralo.

Con el comando `gpg -d` desciframos el fichero.



```

raul@raul-VirtualBox:~$ gpg -d mensaje_cifrado.txt.gpg
gpg: AES256.CFB encrypted data
gpg: cifrado con 1 frase contraseña
Mensaje para cifrar fichero
  
```


- f) ¿Qué algoritmo por defecto usa GPG? Utiliza la opción adecuada para cifrar un fichero de texto con el algoritmo de cifrado TWOFISH. Incluye una captura de pantalla donde se vea tanto el comando utilizado como todas las opciones que aparecen por pantalla hasta generar la clave, y explica tanto los parámetros utilizados en la llamada como todas las opciones que aparecen.

Primero ciframos el fichero con el algoritmo TWOFISH, usando el comando siguiente:

```
raul@raul-VirtualBox:~$ gpg --cipher-algo TWOFISH -c mensaje_cifrado.txt
El fichero 'mensaje_cifrado.txt.gpg' ya existe. ¿Sobreescribir? (s/N) s
```

Cuando ejecutamos el comando se nos abre un pop-up para introducir la contraseña:



Luego para descifrarlo ejecutaremos el siguiente comando:

```
raul@raul-VirtualBox:~$ gpg -d mensaje_cifrado.txt.gpg
gpg: TWOFISH.CFB encrypted data
gpg: cifrado con 1 frase contraseña
Mensaje para cifrar fichero
```

Tercera Parte (nota máxima B): cifrado asimétrico

En esta tercera parte de la práctica vas a crear tu par de claves pública y privada, mediante GnuPG y las usarás para cifrar y descifrar asimétricamente un fichero.

Ejercicio 3. Generar claves

- a) Comprueba mediante el comando `gpg` si tienes alguna clave pública/privada.

```
raul@raul-VirtualBox:~$ gpg --list-keys
raul@raul-VirtualBox:~$ gpg --list-secret-keys
```

No tenemos ninguna clave creada

- b) Genera un par de claves con el comando `gpg`.

Usamos el comando `gpg --full-generate-key`

```
raul@raul-VirtualBox:~$ gpg --full-generate-key
gpg (GnuPG) 2.4.4; Copyright (C) 2024 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Por favor seleccione tipo de clave deseado:
(1) RSA and RSA
(2) DSA and Elgamal
(3) DSA (sign only)
(4) RSA (sign only)
(9) ECC (sign and encrypt) *default*
(10) ECC (sólo firmar)
(14) Existing key from card
Su elección: 1
Las claves RSA pueden tener entre 1024 y 4096 bits de longitud.
¿De qué tamaño quiere la clave? (3072) 1024
El tamaño requerido es de 1024 bits
Por favor, especifique el periodo de validez de la clave.
0 = la clave nunca caduca
<n> = la clave caduca en n días
<n>w = la clave caduca en n semanas
<n>m = la clave caduca en n meses
<n>y = la clave caduca en n años
¿Validez de la clave (0)? 0
La clave nunca caduca
¿Es correcto? (s/n) s
```

```
GnuPG debe construir un ID de usuario para identificar su clave.
Nombre y apellidos: raul
Dirección de correo electrónico:
Comentario:
Ha seleccionado este ID de usuario:
"raul"

¿Cambia (N)ombre, (C)omentario, (D)irección o (V)ale/(S)alir? v
Es necesario generar muchos bytes aleatorios. Es una buena idea realizar
alguna otra tarea (trabajar en otra ventana/console, mover el ratón, usar
la red y los discos) durante la generación de números primos. Esto da al
generador de números aleatorios mayor oportunidad de recoger suficiente
entropía.
Es necesario generar muchos bytes aleatorios. Es una buena idea realizar
alguna otra tarea (trabajar en otra ventana/console, mover el ratón, usar
la red y los discos) durante la generación de números primos. Esto da al
generador de números aleatorios mayor oportunidad de recoger suficiente
entropía.
gpg: creado el directorio '/home/raul/.gnupg/openpgp-revocs.d'
gpg: certificado de revocación guardado como '/home/raul/.gnupg/openpgp-revocs.d/AC403F734434D2F158EC7B5BFB347F3D643F6EA8.rev'
claves pública y secreta creadas y firmadas.

pub   rsa1024 2024-06-16 [SC]
      AC403F734434D2F158EC7B5BFB347F3D643F6EA8
uid    raul
sub    rsa1024 2024-06-16 [E]
```


- c) Comprueba que se han generado.

Usamos los comandos del primer apartado para comprobar las claves

```
raul@raul-VirtualBox:~$ gpg --list-keys
gpg: comprobando base de datos de confianza
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: nivel: 0  validez: 1  firmada: 0  confianza: 0-, 0q, 0n, 0m, 0f, 1u
/home/raul/.gnupg/pubring.kbx
-----
pub   rsa1024 2024-06-16 [SC]
      AC403F734434D2F158EC7B5BFB347F3D643F6EA8
uid   [ absoluta ] raul
sub   rsa1024 2024-06-16 [E]

raul@raul-VirtualBox:~$ gpg --list-secret-keys
/home/raul/.gnupg/pubring.kbx
-----
sec   rsa1024 2024-06-16 [SC]
      AC403F734434D2F158EC7B5BFB347F3D643F6EA8
uid   [ absoluta ] raul
ssb   rsa1024 2024-06-16 [E]
```

Ejercicio 4. Enviar clave pública

Ahora vas a enviar la clave pública a otro compañero por correo-e. Lo más frecuente es subir la clave pública a un servidor de claves (<https://www.rediris.es/keyserver/index.html.es>) pero nosotros lo haremos así para simplificar: Ponte de acuerdo con algún compañero del aula a través del foro.

- a) Exporta tu clave pública a un fichero.

Exportamos la clave publica

```
raul@raul-VirtualBox:~$ gpg --export -a raul > clave_publica.asc
raul@raul-VirtualBox:~$ ls -l
total 52
-rw-rw-r-- 1 raul raul 43 jun 16 17:45 archivo_practica.txt
-rw-rw-r-- 1 raul raul 1010 jun 16 18:31 clave_publica.asc
```

- b) Envía la clave pública a tu compañero por correo.

Ejercicio 5. Importar claves públicas

- a) Importa la clave pública de tu compañero.
b) Comprueba que se ha importado la clave a tu anillo de claves.

- c) Crea un fichero que se llame `fichero_ejemplo_ejer_5.txt` y encriptalo con la clave que acabas de importar.
- d) Envía por correo-e el fichero a tu compañero, encriptado con su clave pública.

Ejercicio 6. Desencriptar el fichero que os han enviado

- a) Comprueba que no puedes ver el contenido del fichero que te han enviado.
- b) Desencripta el fichero.
- c) Comprueba que ahora puedes ver el contenido del fichero que te han enviado encriptado con tu clave pública.

Ejercicio 7. Otras opciones de GPG

- a) Indica qué comando usarías para borrar una clave de tu anillo de claves.

Usaría el siguiente comando para borrar la clave

```
raul@raul-VirtualBox:~$ gpg --delete-key-key "raul"  
gpg: invalid option "--delete-key-key"
```

- b) Indica cómo firmarías un mensaje, sin encriptarlo.

Usaría el siguiente comando para firmar el mensaje sin encriptarlo

```
gpg: invalid option --delete-key-key  
raul@raul-VirtualBox:~$ gpg --clear-sign mensaje_cifrado.txt
```

Cuarta Parte (nota máxima A): configuración segura de Apache, conexión con cliente y explicación de SSL/TLS

Apache, disponible tanto para Linux como para Windows, es el segundo servidor HTTP más utilizado hoy en día (<https://w3techs.com/>). Está diseñado con una estructura modular de tal manera que, además del servidor, existen diferentes módulos que permiten añadirle varias capacidades.

En esta parte habilitaremos el módulo SSL de Apache, que ofrece soporte para SSL v3 y TLS v1.x, y configuraremos el HTTPS para tener un servidor web seguro. Después realizaremos la conexión al servidor con varios clientes. En primer lugar, usaremos la propia herramienta `openssl` en modo cliente y posteriormente lo haremos con un navegador web.

Habilitar el módulo `mod_ssl`

El módulo `mod_ssl` de Apache2 proporciona la capacidad de encriptar los datos mediante SSL (*Secure Socket Layer*). Cuando nos queremos conectar a un sitio web con estas características, hay que usar el prefijo `https://` en la URL, en la barra de direcciones del navegador. SSL está basado en criptografía de clave pública (PKI).

El módulo `mod_ssl` viene instalado por defecto con el Apache2. Para habilitarlo se usa la orden `a2enmod` (*Apache2 Enable Module*). Por ejemplo, lo siguiente habilitaría el módulo `mod_ssl`:

```
sudo a2enmod ssl
```

Para que los cambios se apliquen tienes que recargar la configuración de la Apache2 o reiniciarlo mediante la orden:

```
sudo service apache2 restart
```

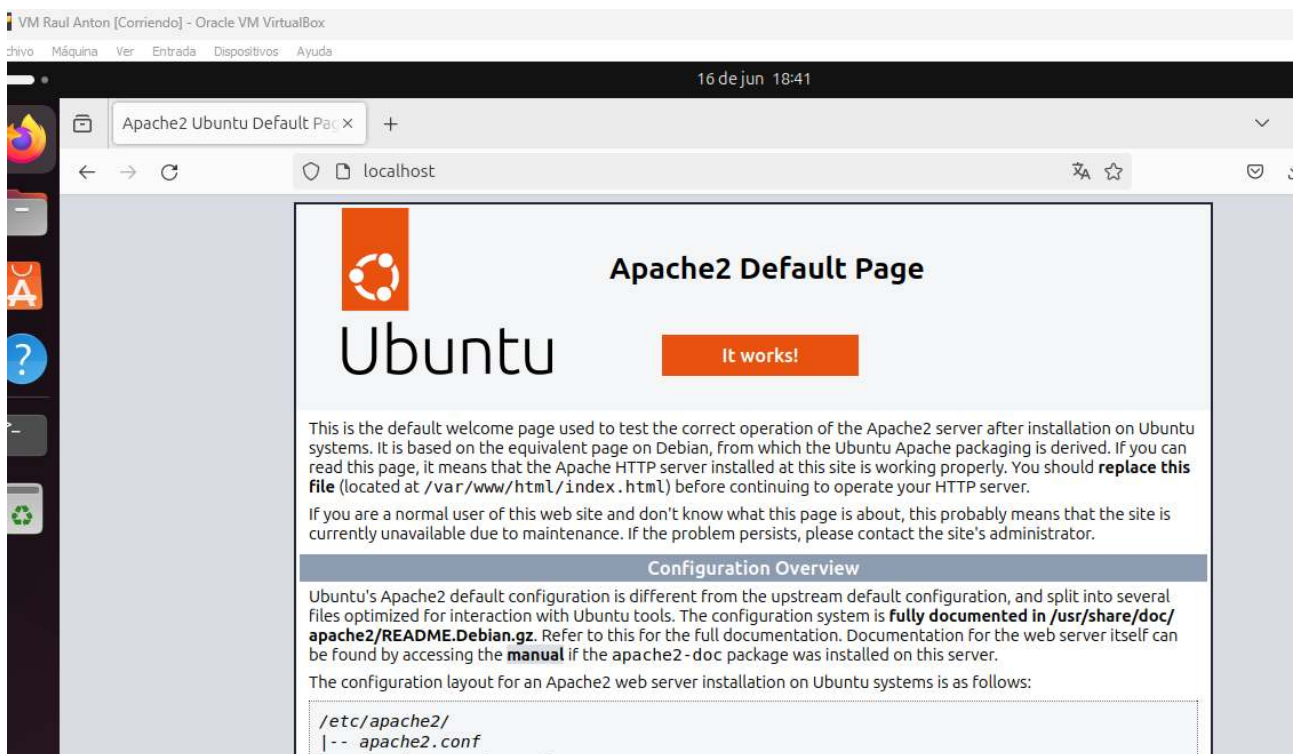
Nota.- Para instalar cualquier módulo de Apache2 puedes usar la herramienta `apt-get`.

Ejercicio 8. Configuración de HTTPS

Apache ya trae un certificado autofirmado que podemos usar en nuestros sitios web, así como un fichero de configuración para SSL llamado `default-ssl`.

- a) Instala Apache mediante `apt` y comprueba que funciona por HTTP.

```
raul@raul-VirtualBox:~$ sudo apt-get install apache2
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
 apache2-bin apache2-data apache2-utils libapr1t64 libaprutil1-dbd-sqlite3 libaprutil1-ldap libaprutil1t64
Paquetes sugeridos:
 apache2-doc apache2-suexec-pristine | apache2-suexec-custom
Se instalarán los siguientes paquetes NUEVOS:
 apache2 apache2-bin apache2-data apache2-utils libapr1t64 libaprutil1-dbd-sqlite3 libaprutil1-ldap libaprutil1t64
0 actualizados, 8 nuevos se instalarán, 0 para eliminar y 40 no actualizados.
Se necesita descargar 1.896 kB de archivos.
Se utilizarán 7.452 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s
Des:1 http://es.archive.ubuntu.com/ubuntu noble/main amd64 libapr1t64 amd64 1.7.2-3.1build2 [107 kB]
Des:2 http://es.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1t64 amd64 1.6.3-1.1ubuntu7 [91,9 kB]
Des:3 http://es.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1-dbd-sqlite3 amd64 1.6.3-1.1ubuntu7 [11,2 kB]
Des:4 http://es.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1-ldap amd64 1.6.3-1.1ubuntu7 [9,116 B]
Des:5 http://es.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2-bin amd64 2.4.58-1ubuntu8.1 [1.327 kB]
Des:6 http://es.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2-data all 2.4.58-1ubuntu8.1 [163 kB]
Des:7 http://es.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2-utils amd64 2.4.58-1ubuntu8.1 [96,2 kB]
Des:8 http://es.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2 amd64 2.4.58-1ubuntu8.1 [90,2 kB]
Descargados 1.896 kB en 0s (3.839 kB/s)
```



Instalamos apache y comprobamos q funcione.

- b) Cambia la página por defecto para que muestre “El servidor de <tu nombre y apellidos> funciona”.

Editamos el html y comprobamos.

```
GNU nano 7.2 /var/www/html/index.html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org
<html xmlns="http://www.w3.org/1999/xhtml">
<!--
  Modified from the Debian original for Ubuntu
  Last updated: 2022-03-22
  See: https://launchpad.net/bugs/1966004
-->
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <title>El servidor de Raúl Antón González</title>
  <style type="text/css" media="screen">
  * {
    margin: 0px 0px 0px 0px;
    padding: 0px 0px 0px 0px;
  }

  body, html {
    padding: 3px 3px 3px 3px;

    background-color: #D8DBE2;

    font-family: Ubuntu, Verdana, sans-serif;
    font-size: 11pt;
    text-align: center;
  }
  }
```



- c) Localiza en el fichero `/etc/apache2/sites-available/default-ssl` las directivas que tienen que ver con SSL y explica su significado.


```
# SSL Engine Switch:
# Enable/Disable SSL for this virtual host.
SSLEngine on

# A self-signed (snakeoil) certificate can be created by installing
# the ssl-cert package. See
# /usr/share/doc/apache2/README.Debian.gz for more info.
# If both key and certificate are stored in the same file, only the
# SSLCertificateFile directive is needed.
SSLCertificateFile    /etc/ssl/certs/ssl-cert-snakeoil.pem
SSLCertificateKeyFile  /etc/ssl/private/ssl-cert-snakeoil.key

# Server Certificate Chain:
# Point SSLCertificateChainFile at a file containing the
# concatenation of PEM encoded CA certificates which form the
# certificate chain for the server certificate. Alternatively
# the referenced file can be the same as SSLCertificateFile
# when the CA certificates are directly appended to the server
# certificate for convinience.
#SSLCertificateChainFile /etc/apache2/ssl.crt/server-ca.crt

# Certificate Authority (CA):
# Set the CA certificate verification path where to find CA
# certificates for client authentication or alternatively one
# huge file containing all of them (file must be PEM encoded)
# Note: Inside SSLCACertificatePath you need hash symlinks
#       to point to the certificate files. Use the provided
#       Makefile to update the hash symlinks after changes.
```

- SSLEngine: Habilita o deshabilita el motor SSL para el host virtual.
- SSLCertificateFile: Especifica la ruta al archivo del certificado SSL.
- SSLCertificateKeyFile: Especifica la ruta al archivo de la clave privada del certificado SSL.
- SSLCertificateChainFile: Especifica la ruta al archivo de la cadena de certificados.
- SSLCACertificatePath y SSLCACertificateFile: Especifican la ruta a los certificados de la Autoridad Certificadora.

d) Habilita HTTPs tecleando:

```
sudo a2ensite default-ssl
```

```
raul@raul-VirtualBox:~$ sudo a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
systemctl reload apache2
```

e) Comprueba mediante la orden `netstat -ntln` que HTTPs está habilitado.

```
Procesando dispositivos para non-ss (27.12.10-4ubuntu2) ...
raul@raul-VirtualBox:~$ sudo netstat -ntln
Conexiones activas de Internet (solo servidores)
Proto Recib Enviad Dirección local Dirección remota Estado
tcp 0 0 127.0.0.53:53 0.0.0.0:* ESCUCHAR
tcp 0 0 127.0.0.54:53 0.0.0.0:* ESCUCHAR
tcp 0 0 127.0.0.1:631 0.0.0.0:* ESCUCHAR
tcp6 0 0 :::631 :::* ESCUCHAR
tcp6 0 0 :::80 :::* ESCUCHAR
tcp6 0 0 :::443 :::* ESCUCHAR
```


Ejercicio 9

Conéctate a tu servidor utilizando la siguiente orden:

```
openssl s_client -connect 127.0.0.1:443
```

```

Start Time: 1718560912
Timeout   : 7200 (sec)
Verify return code: 0 (ok)
Extended master secret: no
Max Early Data: 0
---
read R BLOCK
---
Post-Handshake New Session Ticket arrived:
SSL-Session:
    Protocol  : TLSv1.3
    Cipher    : TLS_AES_256_GCM_SHA384
    Session-ID: B9F9AF7F47CB0C4F4E229D0005E4E2CD354FB7F9C50CBA365FEE5BFB974782A8
    Session-ID-ctx:
    Resumption PSK: 414D8D321C870ED98A6D2B705846AE69A5C648AD8C2ED0BE3096BA85B339222F8C55F55B3C0A9B35BE353DE8DD663E2D
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 300 (seconds)
    TLS session ticket:
    0000 - c6 50 dd 18 b5 c5 b7 54-7a bc 81 f3 c0 87 f3 ce  .P.....Tz.....
    0010 - f7 9b 68 25 50 a7 8d c4-0d 85 15 b2 f0 1f 8e 00  ..h%P.....

Start Time: 1718560912
Timeout   : 7200 (sec)
Verify return code: 0 (ok)
Extended master secret: no
Max Early Data: 0
---
read R BLOCK

```

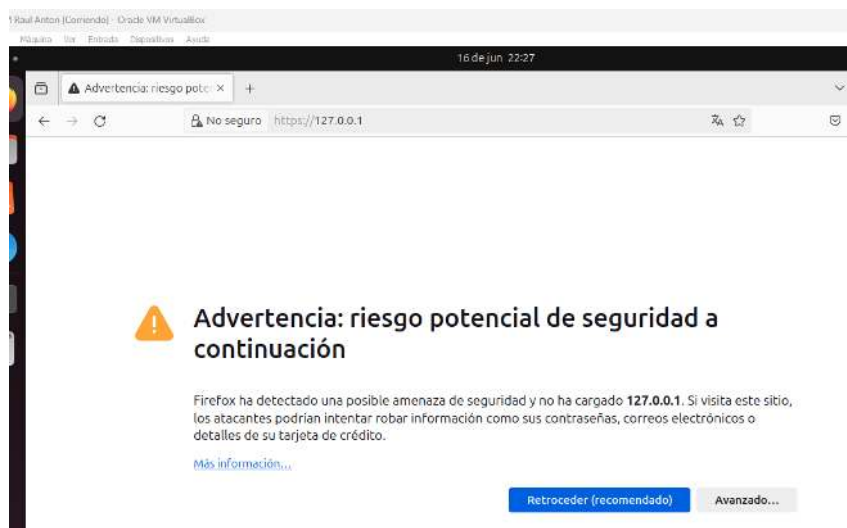
Explica **en detalle** qué significado tienen los campos siguientes dentro de SSL-Session:

- Protocol: Es la versión del protocolo SSL que se utiliza para la conexión
- Cipher: Muestra el cifrado que se está utilizando para la conexión. Define el tipo de encriptación de los datos que vane entre el cliente y el servidor.
- Session-ID: Es un identificador único para la sesión SSL.
- TLS session ticket: Es un ticket de sesión TLS, que es una forma de mantener la información de la sesión sin tener que mantener un estado en el servidor.

Ejercicio 10

Realiza una conexión SSL con tu navegador contra tu servidor mientras capturas el tráfico con el Wireshark (utiliza el modo puente del VirtualBox):

- a) **Pega una captura de pantalla** donde se vea la advertencia de tu navegador web al conectarse a un servidor con un certificado autofirmado.



- b) Explica qué algoritmos se han usado en la conexión.

Se ha usado un algoritmo de intercambio de claves, como el ECDHE, también un algoritmo de sesión AES y un hash SHA256.

- c) Identifica y explica los tres pasos del *SSL handshake* en la captura.

- 1- **ClientHello**: Es cuando el cliente inicia la conexión SSL enviando un mensaje ClientHello al servidor. Donde incluye las versiones del SS compatibles y otros parámetros.
- 2- **ServerHello**: El servidor responde con un ServerHello que selecciona los parámetros de la conexión, como la versión SSL.
- 3- **Client Key Exchange**: El cliente genera una clave, la cifra con la pública del servidor y la envía al servidor. El servidor descifra la clave con su clave privada. Entonces ambos lados generan claves de sesión que se utilizarán para cifrar los datos. Una vez hecho el cliente y el servidor envían un mensaje Finished, indicando que el handshake terminó.

Referencias

- <https://www.openpgp.org/>
- <https://www.gnupg.org/> • <https://www.apache.org>