Usando GPG

Maximiliano A. Christener

1. Creando llaves

Para crear un par de llaves, escribimos en la terminal

```
gpg --full-generate-key
```

El comando nos lleva a un asistente, que consiste en

1. `(1) RSA y RSA (por defecto)`
2. `¿De qué tamaño quiere la clave?` Más bits, mejor encriptación.[1]
3. `Por favor, especifique el período de validez de la clave.`
4. `Nombre y apellido` Identificación.
5. `Dirección de correo electrónico` Idem.
6. `Comentario` Opcional.
7. `Contraseña` Que sea buena y única.
8. `Es necesario generar muchos bytes aleatorios...`[2]

Con eso, las claves fueron creadas.

[1] Por lo cual, escoger el máximo número es la mejor opción.

[2] Sugiero hacerle caso a la terminal: Realizar otras tareas demandantes para que el procesador genere entropía de manera que se pueda mejorar la encriptación.

2. Exportando e importando

Para verlas, utilizamos

```
gpg --list-keys
```

Nos devuelve una salida poco familiar

```
pub rsa1024 2022-08-14 [SC] [caduca: 2022-08-15]
4555E7FE991AAE8FA3C159E69DA89577C6E26A31
uid [ absoluta ] holaa testtt <test@ing.com>
sub rsa1024 2022-08-14 [E] [caduca: 2022-08-15]
```

La serie `4555E7FE991AAE8FA3C159E69DA89577C6E26A31` es la *fingerprint* de nuestro par de llave. Es importante tenerlo en cuenta. En `uid` aparece el nombre y correo que estemos usando, el cual también va a ser expuesto en nuestra llave pública.

Ahora, estamos interesados en tener bajo control estas llaves, y poder hacernos responsables de las mismas. Para hacerlo con nuestra llave **pública**, utilizamos el siguiente comando

```
gpg --export -a fingerprint >public.asc
```

Donde `--export` significa que queremos exportar nuestra clave pública, `-a` es la opción de hacerlo en formato ASCII, luego ponemos nuestra key ID y por último `public.asc` es el archivo donde lo vamos a guardar[3].

El análogo para la clave **privada** es

```
gpg --export-secret-keys -a fingerprint >private.asc
```

Para **importar** alguna clave desde un archivo `.asc`, nos dirigimos al directorio, y simplemente hacemos

```
gpg --import file.asc
```

Donde `file.asc` es el archivo donde se encuentra la llave pública o privada[4]

[3] El nombre es arbitrario, se puede escribir cualquiera.

[4] Al importar una llave privada, automáticamente aparece la pública. Es decir, privada *implica* pública, pero nunca viceversa.

Certificado de revocación Para crear un `.asc` del mismo, se utiliza el siguiente comando

```
gpg --gen-revoke FINGERPRINT > revokefile.asc
```

Una vez más, `revokefile.asc` se trata del fichero donde se encuentra el certificado de revocación[5]. Tener mucha precuación con él, así como con la llave privada.

3. Servidor de llaves

Es deseable enviar las claves públicas a un servidor de llaves. Para ello

```
gpg --send-keys fingerprint
```

Notar que, si no especificamos el servidor, `GnuPG` va a enviar las llaves a `hkps://keys.openpgp.org`. Si no queremos que eso ocurra[6], entonces habrá que agregar la opción `--keyserver link` y especificar alguno de preferencia.

Utilizando el comando por segunda vez Haciéndolo, envía al servidor nuestro *certificado de revocación*, por si queremos actualizar o reemplazar nuestra clave.

3.1. Uso con `keys.openpgp.org`

Dado que el comando anterior no verifica la identidad del propietario del correo electrónico, una medida de seguridad debe ser enviar un correo electrónico de verificación para poder ser publicado. Una forma rápida de conseguir el link es haciendo el proceso descripto pero utilizando este comando en su lugar

```
gpg --export MAIL  curl -T - https://keys.opengpg.org
```

Este es el servidor que recomiendo dada su practicidad.

3.2. Uso con `pgp.mit.edu`

Para este, es mejor entrar directamente al sitio web y leer los pasos.

[5]Se puede controlar con el comando `cat`.

[6]Por ejemplo, si deseamos que nuestras claves se encuentren alojadas en `pgp.mit.edu`.

4. Cifrando archivos

Con `GnuPG` y una clave *pública*, podemos cifrar cualquier archivo para un dispositivo que posea la *privada*. Un ejemplo sería

```
gpg -r test@ing.com --output salida.gpg -e test.asc
```

A saber

- `-r` viene de *recipe*, el recipiente donde vamos a cifrar. Luego de esto, debemos introducir el correo electrónico de la llave pública.
- `--output` es, lógicamente, la salida. Es un paso opcional, pero si no lo encaramos, el formato del fichero va a quedar expuesto[7] .
- `-e` para encriptar el archivo, detallado a continuación (test.asc para el ejemplo).

Ahora, *con la llave privada*, se puede desencriptar el archivo usando

```
gpg -d salida.gpg
```

Donde `-d` significa desencriptar.

Notar que, sin llave privada, **no** podemos desencriptar el archivo. Es imposible, no importa si lo hayamos hecho nosotros mismos.

Tener en cuenta que, de todos modos, poseemos el fichero original sin encriptar a nuestra disposición, con lo que, tranquilamente podemos encriptar archivos para otras personas o dispositivos con total tranquilidad.

Si alguno lo quiere intentar por curiosidad, se va a encontrar con el siguiente error

(terminal) `gpg: descifrado fallido: no secret key`

4.1. Usando tar y cifrado simétrico

Hasta ahora, sólo utilizamos la versión asimétrica, es decir, la que requiere una clave pública y otra privada —por lo tanto, la más segura[8]—. A veces, es conveniente usar sólo la simétrica, la que sólo pide contraseña.

Primero, en caso de ser necesario, creamos un `tar` de nuestro directorio

[7] Por ejemplo, en nuestro caso, tendríamos test.asc.gpg como salida predeterminada.

[8] ≪Segura≫ en el sentido que el método está ideado para garantizarle la identidad al sistema. Evidentemente, lidiar con algo tan sensible como lo es la clave privada es en ocasiones poco práctico, razón por la cual existe esta alternativa.

```
tar -cf nombre.tar DIRECTORIO
```

Donde `c` es para crear un tarball, y `f` para especificar el archivo a escribir.
Para comprimir, usamos

```
xz nombre.tar
```

Si usamos -9, añadimos la máxima compresión.
Para **cifrar** simétricamente

```
gpg -c nombre.tar.xz
```

También podemos usar `--symmetric` en lugar de `-c`. Tras insertar la frase contraseña, tenemos cifrado nuestro archivo.

4.1.1. Desencriptado y descompresión

Para **desencriptar**, tenemos[9]

```
gpg -o nombre.tar.xz -d nombre.tar.xz.gpg
```

Para **descomprimir**, usamos

```
tar -xf nombre.tar.xz
```

5. Eliminando

Para borrar una clave pública, usamos

```
gpg --delete-key FINGERPRINT
```

En caso de ser además propietario de una clave privada, va a aparecer un error, que nos dice que *primero* ejecutemos este comando

```
gpg --delete-secret-key FINGERPRINT
```

Ahora si, ya podemos eliminar la clave pública.

[9]La opción `-o` es el archivo de salida, ya sin encriptar. En el caso de `-d`, simplemente sirve para indicar lo que se va a desencriptar.