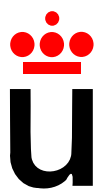
Herramientas: GPG, OpenSSL

Enrique Soriano

LS, GSYC

5 de febrero de 2018



Universidad
Rey Juan Carlos

GPG: generación de claves

- GPG puede tener distinta configuración dependiendo de la distribución (algoritmos por defecto, etc).
- La configuración está en el directorio `$HOME/.gnupg`,
- Para ver los algoritmos soportados, podemos ejecutar:

```
gpg --verbose --version
```

GPG: almacenamiento de claves

- GPG crea varios pares de claves (principal y subordinadas para cifrar) cuando se ejecuta:

  ```
  gpg --gen-key
  ```

- Intenta crear las claves más fuertes: la transición de una clave vieja a una nueva es dura[1].
- Nos dan opciones para elegir:
  - Algoritmo (RSA, Elgamal, DSA (sólo firma), DSA+Elgamal).
  - Longitud clave.
  - Nombre completo.
  - Correo electrónico.
  - Passphrase.

[1] http://ekaia.org/blog/2009/05/10/creating-new-gpgkey/

GPG: almacenamiento de claves

- Las claves subordinadas están ligadas a la clave maestra.
- La clave maestra sólo se usa para firmar (otras claves, etc.). NO se usa para cifrar.
- Hace más cómoda la gestión de claves: se pueden revocar las subclaves y crear otras nuevas.
- Todas las claves públicas van al anillo de claves públicas. Todas las claves privadas van al anillo de claves privadas.

```
$HOME/.gnupg/secring.gpg
$HOME/.gnupg/pubring.gpg
```

GPG: almacenamiento de claves

- La passphrase se usa para sacar una clave mediante una KDF llamada s2k.
- El anillo privado se cifra con esa clave, con un algoritmo de clave simétrica, CAST5, en modo CFB.
- Para ver las claves privadas:

  `gpg --list-secret-keys`
- Para ver las claves que tenemos en el anillo público:

  `gpg --list-keys`
  - `pub`: clave pública, con la longitud de clave, algoritmo (D=DSA, G=ElGamal, R=RSA), KeyID de clave,, fecha de creación y fecha de caducidad.
  - `sub`: clave subordinada, mismos atributos.
  - `uid`: datos del propietario (nombre, correo, comentarios). Una clave puede tener más de un UID.
  - `expires`: fecha de vencimiento. A partir de esa fecha ya no se debe usar la clave. El dueño puede extender la fecha de vencimiento cuando quiera. Es deseable establecerlo: medida de *hombre muerto*.

GPG: exportar/importar claves

Para exportar tu clave pública:

```
gpg --output public.asc --armor --export pepe@a.com
```

- `--armor` genera los datos aplanados en Base 64.

Para importar la clave pública de alguien:

```
gpg --import public.asc
```

GPG: exportar/importar claves

Para subir la clave a un servidor de PGP:

```
gpg --keyserver pgp.rediris.es --send-keys 870AE0C4
```

Para bajar la clave pública de alguien desde un servidor:

```
gpg --keyserver pgp.rediris.es --recv-key 870AE0C1
```

Para borrar una clave:

```
gpg --delete-key pepe@pepe.com
```

Si no se indica servidor, se usa el servidor gpg por omisión.

GPG: exportar/importar claves

Para revocar una clave nuestra, debemos generar antes el certificado de revocación:

```
gpg --output revoke.asc --gen-revoke D0A52B5D
```

- Se recomienda hacer esto al crear la clave.
- Sin el certificado, no se podrá revocar una clase.
- El certificado se tiene que guardar en un lugar seguro, por si en el futuro necesitamos revocar la clave.

Para revocar la clave en local y en el servidor:

```
gpg --import revoke.asc
gpg --send-keys --keyserver pgp.rediris.es D0A52B5D
```

GPG: confianza y firma de claves

Para editar una clave pública:

```
gpg --edit-key pepe@gmail.com
```

- Comandos:
  - `fpr` muestra el fingerprint de la clave.
  - `trust` asigna el nivel de confianza en el usuario (web of trust).
  - `sign` firma dicha clave con nuestra clave privada.
  - `check` muestra las firmas que lleva la clave. Todas las claves están autofirmadas, y además pueden estar firmadas por otros usuarios.

GPG: confianza y firma de claves

- Firmar la clave supone dar fe de que dicha clave pertenece a la identidad. Esto sólo se puede hacer si estamos completamente seguros de ello.
- Si firmas una clave en tu anillo, pasa a ser válida.
- Una clave no firmada por ti será valida si reune el número mínimo de firmas con confianza marginal o completa.
- Una vez firmada, puedes dársela a su dueño para que él la distribuya firmada.
- No debes publicar la clave pública firmada sin el permiso de su propietario.

GPG: confianza y firma de claves

Ajustar las restricciones sobre la validez de las claves públicas de terceros:

```
gpg --completes-needed N
```

```
gpg --marginals-needed N
```

GPG: cifrado (clave pública)

Cifrado de datos con una clave pública:

```
gpg --output fichero.gpg \
              --encrypt  [-r recipient] fichero
```

- Si no indicamos el destinatario (recipient, por ejemplo el correo electrónico) con la opción -r, nos lo pedirá de forma interactiva.

GPG: descifrado (clave pública)

Descifrado de datos con mi clave privada:

```
gpg --output  fichero  --decrypt  fichero.gpg
```

- Para poder usar la clave privada, debemos introducir la passphrase correspondiente.

GPG: cifrado (clave simétrica)

Para cifrar con un algoritmo de clave simétrica:

```
gpg --output  fichero.gpg  --symmetric fichero
```

- Debemos introducir la passphrase correspondiente para generar la clave simétrica con la KDF. OJO: esa no debe ser la passphrase que protege tu clave privada.

GPG: cifrado (clave simétrica)

Para cifrar con un algoritmo que no sea el algoritmo por omisión:

```
gpg --output fichero.gpg --symmetric \
          --cipher-algo AES256 fichero
```

- El algoritmo por suele ser CAST5. Aquí estamos usando otro: AES con clave de 256.

GPG: firma

Para firmar un fichero, podemos usar:

```
gpg --output data.sig --sign data
```

- El fichero se comprime y después se firma, dejando todo en el mismo fichero de salida.

GPG: firma

Para hacer lo mismo con un fichero en texto plano para meter la firma al final del documento codificada en PEM, podemos usar:

```
gpg --output data.sig --clearsign data
```

- La salida es el fichero en texto concatenado con la firma aplanada. Ojo: **los blancos (espacios, tabs, etc.) no se tienen en cuenta para la firma**

GPG: firma

Para verificar la firma:

```
gpg --verify  data.sig
```

GPG: firma

Se puede conseguir una firma separada de los datos:

```
gpg --output data.sig --detach-sig data
```

GPG: firma

Para verificar la firma separada:

```
gpg --verify  data.sig data
```

GPG: firmar y cifrar

Se pueden mezclar. Por ejemplo, para cifrar y firmar un archivo:

```
gpg -o fichero.gpg --sign --encrypt -r \
            ana@uu.es  fichero
```

El destinatario puede descrifrar y verificar el fichero:

```
gpg -o fichero --decrypt  fichero.gpg
```

GPG: firmar y cifrar

Se pueden crear resúmenes hash:

```
gpg --print-md SHA1 fichero
```

```
gpg --print-md MD5 fichero
```

GPG: transitar a una nueva clave maestra

Para cambiar de clave maestra (p. ej. porque sea muy vieja):

1. Crear nuevas claves y su certificado de revocación.
2. Firmar las claves nuevas con la clave antigua.
3. Crear una declaración (statement) que explique el cambio de clave, y firmarlo con la vieja y con la nueva. Publicar el documento doblemente firmado.
4. Volver a firmar las claves de otros con la clave nueva (las que sigan activas).
5. Distribuir la clave nueva (conocidos, servidores).
6. Revocar la clave vieja cuando haya pasado un tiempo razonable. Recuerda: la revocación no se puede deshacer.

OpenSSL: crifrado

- También podemos usar OpenSSL para cifrar usando distintos algoritmos.
- Podemos ver la lista de algoritmos de cifrado soportados con

  `openssl --ciphers`
- El flag -a permite trabajar con la entrada/salida aplanada en base64.

OpenSSL: crifrado

Para cifrar con algoritmos de clave simétrica:

```
openssl aes-256-cbc -in fichero -out fichero.ssl
```

Para descifrar:

```
openssl aes-256-cbc -d  -in fichero.ssl -out fichero
```

- Se pedirá una contraseña para sacar la clave simétrica.
- `-nosalt` si no queremos que se use *salt*.

OpenSSL: hash

Para obtener una hash:

```
openssl  dgst  -sha1 fichero
```

OpenSSL: generación de certificados X.509

Para crear un certificado X.509 autofirmando, aplanado en PEM y comprimido:

```
openssl req -new -nodes -x509 -newkey 4096 \
   -out cert.pem  -keyout privkey.pem -days 500 \
   -subj "/C=DE/ST=NRW/L=Earth/O=Random\
          Company/OU=IT/CN=lsub.org/\
          emailAddress=dont@mail.me"
```

OpenSSL: generación de certificados X.509

Si lo queremos en texto plano:

```
openssl x509 -in cert.pem -inform PEM -text
```

Si queremos extraer la clave pública:

```
openssl x509 -inform pem -in cert.pem -pubkey \
                              -noout > pubkey.pem
```

OpenSSL: generación de certificados X.509

Crear un PKCS12 con el certificado y la clave privada:

```
openssl pkcs12 -export -in cert.pem \
   -inkey privkey.pem \
   -out cert.p12 -name  alias-que-quieras
```

OpenSSL: generación de certificados X.509

Si queremos ver qué certificados manda un servidor cuando intentamos establecer una conexión SSL:

```
openssl s_client -showcerts -servername prueba.es \
    -connect prueba.es:1445  </dev/null
```

OpenSSL: comprobación de certificados X.509

Para ver el fingerprint de un certificado:

```
openssl x509 -in cert.pem  -noout -sha1 -fingerprint
```

OpenSSL: verificación de certificados X.509

Verificar una cadena de certificados

```
openssl verify -verbose -CAfile \
   <(cat  CA.pem intermediate1.pem intermediate2.pem) \
   cert.pem
```

OpenSSL: crear claves RSA

```
openssl genrsa -out privkey.pem 4096
openssl rsa -in privkey.pem -out \
            pubkey.pem -outform PEM -pubout
```

- Se pueden crear las claves sin crear un certificado X.509.
- Las claves privada/pública se crean por separado.

OpenSSL: CSR

Si queremos crear un fichero CSR (certificate signing request) para que otro nos firme un certificado para nuestra clave (hay que proporcionar los campos para el certificado):

```
openssl req -new -sha256 -key privkey.pem \
  -out request.csr
```

Se puede comprobar así:

```
openssl req -noout -text -in request.csr
```

Este fichero se manda a la CA para que te genere el certificado firmado.

OpenSSL: firmas

Firmar un fichero:

```
openssl dgst -sha256 -sign private.pem \
             -out signature fichero
```

Verificar una firma:

```
openssl dgst -sha256 -verify pubkey.pem \
             -signature signature.txt fichero
```

OpenSSL: cifrar con RSA

Cifrar:

```
openssl rsautl -encrypt -inkey pubkey.pem \
                  -pubin -in fichero -out fichero.ssl
```

Descifrar:

```
openssl rsautl -decrypt -inkey privkey.pem \
       -in fichero.ssl  -out fichero
```

- NOTA: **No se pueden cifrar más datos que el tamaño de la clave** (menos el tamaño requerido por el *padding*).

OpenSSL: cifrar y descifrar con RSA + AES

Generar clave AES:

```
openssl rand -base64 32 -out aesKey
```

Cifrar:

```
openssl enc -aes-256-cbc -salt -in fichero \
        -out fichero.ssl -pass file:./aesKey.txt
openssl rsautl -encrypt -inkey pubkey.pem \
      -pubin -in aesKey.txt -out aesKey.rsa
```

Descifrar:

```
openssl rsautl -decrypt -inkey privkey.pem -in \
        aesKey.rsa -out aesKey.txt
openssl enc -d -aes-256-cbc -in fichero.ssl  -out \
       fichero  -pass file:./aesKey.txt
```

Otras herramientas

- `shasum` nos permite crear hashes SHA-1.
- `sha256sum`, `sha384sum`, `sha512sum` nos permite crear hashes SHA-2.