“GNU Privacy Guard” (GnuPG) Mini Como

Michael Fischard v. Mollard <*fischer@math.uni-goettingen.de*> (versión en alemán)
Brenno J.S.A.A.F. de Winter <*brenno@dewinter.com*> (versión en inglés)
Horacio <*homega@ciberia.es*> (versión en castellano)

v0.1.3, septiembre de 1999

Este documento trata sobre la instalación, configuración y uso de `Gnu Privacy Guard` (`GnuPG`), un sistema de codificación de código libre y desarrollo abierto, compatible con `OpenPGP`. Con el fin de mantener este programa totalmente libre, se ha evitado el uso de algoritmos con patentes propietarias restrictivas, como las de IDEA y RSA. El documento original fue escrito en alemán por Michael Fischard v. Mollard, y posteriormente traducido, y revisado en algunos puntos, al inglés por Brenno J.S.A.A.F. de Winter. La traducción de este documento al castellano se llevó a cabo desde la versión inglesa. El capítulo 5 ha sido añadido en la versión en castellano, y también se han incluido algunos recursos y otra información en castellano. Esta versión es una revisión de la versión 0.1.2, y no incluye ninguna temática nueva, tan sólo su conversión de código HTML a código SGML para su posterior reconversión a otros formatos. También se han corregido algunos errores de forma y/o traducción.

Contents

1 Conceptos Básicos

1.1 Sistemas de Claves Públicas

Para poder entender mejor el sistema de codificación usado por los sistemas de claves asimétricas (ie. claves públicas y privadas), es necesario entender las diferencias con los sistemas de claves simétricas (ie. claves secretas).

Los sistemas de cifrado con *clave simétrica* son aquéllos en los que la clave que se usa para cifrar una serie de datos, es la misma que la que se usará para descifrar estos datos. En el caso del correo electrónico, el remitente cifraría el mensaje con una *clave secreta*, y para que el destinatario pueda descifrarlo, necesitaría haber obtenido previamente esta misma clave de un modo «seguro», o sea de modo que la clave no haya podido ser interceptada durante la entrega. Si no tenemos la completa seguridad de que el intercambio de la clave ha sido seguro, la validez de este sistema es nula.

Por el contrario, los sistemas de cifrado con *claves asimétricas* usan claves distintas para el cifrado y posterior descifrado de los datos. En un caso como el anterior, el remitente usaría la *clave pública* del destinatario para cifrar el mensaje, y el destinatario descifraría el mensaje con su propia *clave privada*. Así pues, la *clave privada* no debe ser accesible para **nadie** que no sea el propio dueño de la misma, mientras que la *clave pública*, puede ser entregada a cualquier persona. En un sistema de cifrado bien implementado, la *clave privada* no debe derivar nunca de la *clave pública*.

1.2 Firmas Digitales

El concepto de la **firma digital** se basa en la verificación de la autoría de un mensaje. Esto quiere decir que se puede comprobar que el destinatario del mensaje puede comprobar que el «supuesto» remitente es quien afirma ser. Para ello, el remitente, una vez compuesto el mensaje, lo firma usando su propia clave privada. El destinatario, una vez ha recibido el mensaje, comprobará la veracidad de éste, esto es, lo verificará usando la clave pública del remitente.

Este método es de especial utilidad para reducir riesgos de seguridad en nuestros sistemas (nos podrían enviar un supuesto parche para un programa, y éste en realidad ser un virus o un troyano); también podrían enviarnos información o datos, como provenientes de una fuente lícita o fiable. En ambos casos, no sería muy difícil falsificar la dirección y nombre del remitente, pero sí imposible falsificar la firma digital de éste.

Como ya hemos dicho, la verificación de un mensaje firmado digitalmente se lleva a cabo mediante el uso de la **clave pública** del remitente **sobre el texto** del propio mensaje. De este modo no sólo podemos verificar la identidad del autor, sino que también podemos comprobar la integridad del mensaje, ya que la firma digital ha sido generada con el **texto** y la **clave privada**. Así pues, una alteración o modificación del texto «a posteriori», o cualquier manipulación del mensaje (especialmente si hacemos uso de las especificaciones MIME/PGP), daría como resultado un error en la verificación.

1.3 Anillos de Confianza

Un punto flaco en los algoritmos de clave asimétrica es la transmisión del código público. Es posible que una persona ponga en circulación código con un identificador de usuario falso. Si se codifican mensajes con este pseudo código, el intruso los puede descodificar y leerlos.

La solución `PGP` (y por consiguiente la solución `GnuPG`) está en firmar los códigos. La clave pública de un usuario puede estar **firmada** con las claves de otros usuarios. El objetivo de estas firmas es el de reconocer que el UID (identificador de usuario) de la clave pertenece al usuario a quien dice pertenecer. A partir de ahí es un problema de cada usuario de `GnuPG` el decidir hasta qué punto se puede fiar de la firma. Una clave se puede considerar fiable cuando se confía en el remitente y cuando se sabe con seguridad que dicha clave pertenece a éste. Sólo cuando se puede confiar plenamente en la clave del firmante, se puede confiar en la firma que acompaña a la clave de un tercero. Para tener la certeza de que la clave es correcta hay que compararla con la **huella digital** por medio de canales fiables (por ejemplo, podríamos buscar el teléfono en la guía y llamarle, y que nos la dijera de palabra para poder compararla), antes de darle una confianza absoluta.

1.4 Límites de Seguridad

Si lo que se desea es mantener la confidencialidad de los datos que se poseen, no basta con determinar qué algoritmo de cifrado se va a usar; también es necesario pensar en la seguridad general del sistema. En principio, `PGP` está considerado como suficientemente seguro, y hasta el momento no se sabe que haya habido ningún incidente en el que una clave PGP haya sido descodificada. Pero eso no significa que todo lo cifrado sea seguro; si la NSA (Agencia de Seguridad Nacional de los EE.UU.) hubiera conseguido descodificar una clave PGP mediante criptoanálisis, analización del código, o cualquier otro modo, no es probable que lo hicieran público. Pero aún en el caso de que las claves PGP fueran a todas luces imposibles de descodificar, otros tipos de ataques a la seguridad pueden ser utilizados. A principios de Febrero fue detectado un troyano que buscaba las claves PGP en el disco duro, y las transfería al atacante mediante FTP. Si en este caso hubiéramos escogido una contraseña débil o fácil, un simple análisis que consistiera en un «ataque de diccionario» la descubriría en poco tiempo.

Otra posibilidad técnica, aunque más difícil, es la de los troyanos que recogen entradas de teclado y las transmiten al asaltante. También es posible, aunque muy difícil, pasar el contenido de una pantalla a otra. En este último caso no sería necesario ningún análisis sobre datos cifrados, ya que se obtendrían «pre-cifrados».

Por todo esto es necesaria una planificación de la seguridad que esté bien prevista y que minimice los riesgos.

La idea no es la de recrear una atmósfera de paranoia entre la gente, sino dejar claro que para implementar un sistema seguro no basta con la instalación de un programa criptográfico, que si bien es un paso hacia un sistema más seguro, no es una solución completa. Troyanos como el aparecido en Marzo de 1999 (Melissa) probaron que muchas compañías no se encuentran preparadas en temas de seguridad.

2 Instalación y Configuración

2.1 Fuentes de GnuPG

El sitio oficial para para la distribución de `GnuPG` es *sitio oficial de GnuPG* . En las páginas oficiales de *GnuPG* también se pueden encontrar enlaces a réplicas oficiales.

Debido a restricciones legales no se permite bajar material criptográfico desde servidores localizados en los EE.UU., a los no residentes en este país. En EE.UU. existen unas leyes que imponen restricciones a la exportación de código criptográfico así como de programas que los incluyan. Por esta razón PGP se encuentra siempre disponible en dos versiones: una internacional y otra para los EE.UU. El código fuente para la versión internacional fue exportado en formato ascii imprimido en un libro. A continuación fue escaneado en Europa (Oslo) y recompuesto. Se puede obtener más información al respecto en la *página internacional de PGP* . La versión internacional de PGP puede ser importada libremente a los EE.UU. siempre y cuando no se vuelva a re-exportar fuera de éstos. Una vez se ha instalado una versión de `GnuPG` o `PGP`, se debería verificar la firma digital que acompaña al fichero (Ver 6 (Firmar y Verificar)).

2.2 Configuración

GnuPG se puede obtener como un paquete de binarios de *Debian* (.deb), como un paquete de binarios de *RedHat* (.rpm), o en código fuente. Los paquetes son archivos comprimidos de los ficheros binarios que se pueden instalar con las correspondientes herramientas, según la distribución. Si se necesita instalar GnuPG en otros sistemas operativos, entonces lo debe compilar uno mismo a partir de los fuentes. Se agradece que quien compile un paquete de instalación para un sistema o arquitectura diferente, lo haga de dominio público.

Dado que la mayoría del desarrollo de GnuPG tiene lugar en máquinas x86 bajo *Linux* , la migración a un sistema diferente no debería ser ningún problema. La lista de sistemas operativos que están soportados por GnuPG se puede encontrar en las páginas de *GnuPG* . El procedimiento que se describe a continuación no es exclusivo de ninguna plataforma. Este procedimiento se puede usar para compilar e instalar GnuPG partiendo de un archivo comprimdo del código fuente (foo.tar.gz).

Descomprimir y desarchivar el paquete con los fuentes con la orden (si estamos usando `Gnu tar`):

```
$ tar zxvf gnupg-?.?.?.tar.gz
```

El siguiente paso es cambiar al directorio que contenga el código fuente, y ejecutar el guión de configuración:

```
$ ./configure
```

Con este paso no debería suceder nada especial; si ejecutamos

```
$ ./configure --help
```

se pueden ver las opciones de configuración que existen para la compilación. Si durante la internacionalización (GET text) ocurriera algún problema, se puede incluir una versión que venga código fuente, usando la opción

```
--with-included-gettext
```

o desactivarla usando la opción

```
--disable-NLS
```

2.3 Compilación

A continuación, para empezar a compilar ejecutaremos la orden de compilación:

```
$ make
```

Esto debería funcionar sin problemas. Si ocurriera algo anormal, se seguirán los siguientes pasos (en el mismo orden en el que se describen aquí): Primero, intentar solucionarlo por uno mismo (haciendo uso de la documentación existente); segundo, asegurarnos de que no es un error conocido (comprobar el fichero BUGS en *http://www.gnupg.org* . Si estos pasos no resuelven el problema, enviar la pregunta a la lista de correo de GnuPG (en inglés) (ver 7 (Fuentes de Información)). Por si el problema estuviera relacionado con la compilación, se debería intentar `make clean`, ejecutar `configure` de nuevo, e intentar otra vez la compilación. Si nada de esto soluciona el problema, es el momento de preocuparse de verdad.

2.4 Instalación

Suponiendo que hayamos compilado el programa sin problemas, el siguiente paso es instalarlo. Para ello ejecutaremos la orden de instalación:

```
$ make install
```

que copiará el programa y las páginas de manual en el directorio de instalación. Si no hemos cambiado el directorio de instalación cuando ejecutamos `./configure`, éste será `/usr/local/share/gnupg/` por defecto. Es posible encontrar este directorio en el fichero `options.skel`. Cuando se cambie `options.skel`, si se copia a `~/.gnupg/options`, se usarán los ajustes típicos. Al crear `~/.gnupg/` la acción copiar debería ser automática. Todas las opciones posibles están bien documentadas, y tratar de explicarlas aquí no sería de utilidad.

Se puede ejecutar GnuPG como suid root. De este modo el programa se ejecutará con todos los permisos que tiene el superusuario, y se excluye la posiblidad de que ciertas partes del programa se guarden externamente y puedan ser leídas por cualquiera. Sin entrar a valorar los riesgos de ejecutar el programa como superusuario, se debería alertar de los peligros que conllevaría un troyano, ya que éstos, si se está ejecutando como superusuario, pueden dañar todo el sistema. Si por esta razón, o por cualquier otra, se decide no ejecutar GnuPG como root, es posible desactivar el aviso activando `no-secmem-warning` en `~/gnupg/options`.

3 Uso y Gestión de las Claves

3.1 Generar una Clave

Con la orden

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

se genera un nuevo par de claves (el par se compone de clave privada y clave pública). La primera pregunta es qué algoritmo se va a usar. Para más información sobre algoritmos, ver *PGP DH vs. RSA FAQ* o 7.5 (Applied Cryptography). El algoritmo recomendado por GnuPG es DSA/ElGamal, ya que éste no está patentado.

La siguiente pregunta es la longitud de la clave. Esta parte depende de los requerimientos del usuario. Es necesario elegir entre la seguridad y el tiempo de los procesos. Cuanto mayor sea una clave, menor será el riesgo de que el mensaje sea descodificado si es interceptado, pero también aumentará el tiempo que empleará para el cálculo de los procesos. El tamaño mínimo que requiere GnuPG es de 768 bits, aunque mucha gente opina que debería ser de 2048 (que es el máximo con GnuPG en este momento). Para DSA 1024 es un tamaño fijo. Cuando la seguridad es una prioridad más alta que el tiempo, la opción es elegir el tamaño de clave más grande que se permita.

El sistema nos pedirá a continuación que se introduzca el nombre, comentario y dirección de correo electrónico. El código será calculado en base a estas entradas. Esto se puede cambiar más tarde (ver 3.5 (Administración de Claves)). La dirección de correo electrónico que se escoja debería ser una válida, ya que ésta será usada para firmar el identificador de usuario. Si esta dirección se modifica en algún modo, la firma no corresponderá. Finalmente, se puede introducir un comentario.

El último paso consiste en introducir una contraseña. Nótese la diferencia entre los términos anglosajones para la palabra «contraseña»: el término "password" indica una "palabra de paso", mientras que el término "passphrase" indica una "frase de paso". Por tanto esta contraseña se debe componer de más de una palabra. Para que una contraseña sea efectiva (segura), deberá contener los siguientes elementos:

- que sea larga;
- que combine mayúsculas, minúsculas y números;
- que contenga carácteres especiales (no alfanuméricos);
- que sea difícil de adivinar. Por lo tanto, que no sean nombres, fechas significativas, números de teléfono, números de documentos, ...

En general, para una contraseña fuerte es aconsejable intercalar maYúsCUlas con mInúsCulas, números, otros carácteres no alfanuméricos, etc. Al escoger las palabras y frases debemos evitar aquellas palabras demasiado obvias, o fechas significativas, y nunca usar citas de libros o frases famosas. Dicho esto, debemos asegurarnos de que la contraseña que elijamos sea lo suficientemente difícil para que no pueda ser traspasada por un «ataque de fuerza bruta», ni siquiera por un «ataque de diccionario», pero lo suficientemente fácil como para que la recordemos. Si olvidáramos una contraseña nuestra clave quedaría totalmente inutilizada, y los criptogramas con ella cifrados, indescifrables. Ante esta posibilidad se recomienda crear siempre certificados de revocación junto con las claves (ver 3.4 (Certificados de Revocación)).

Una vez se han introducido todos los datos requeridos, empieza el proceso de generación de las claves, que tardará un tiempo dependiendo del tamaño de éstas. Durante este proceso el programa recoge datos aleatorios que usará para generar las claves; un modo para ayudar a generar estos datos es cambiando a una consola virtual diferente y usando el teclado mientras el proceso está en marcha.

3.2 Exportar Claves

La orden para exportar la clave es:

```
$ gpg --export [UID]
```

Si no designamos un identificador de usuario (UID) todas las claves presentes en el anillo de claves serán exportadas. El resultado es enviado por defecto a `stdout`, pero con la opción `-o` podemos especificar que sea enviado a un fichero. Se recomienda usar la opción `-a` para que el resultado sea un fichero de 7-bit ASCII en lugar de un fichero binario.

Al exportar la clave pública se amplía el abanico de personas con las que se podrá comunicar de modo seguro. La clave se puede exportar poniéndola en una página *web*, mediante *finger*, *ftp*, subiéndola en un servidor de claves públicas, o cualquier otro método.

3.3 Importar Claves

Cuando se recibe la clave pública de otra persona hay que añadirla a la base de datos (anillo de claves) para poder hacer uso de ella. La orden para importarlas es la siguiente:

```
$ gpg --import [fichero]
```

Si se omite el nombre del fichero se leerán los datos de `stdin`. El fichero puede contener una sola clave o varias a la vez, pertenecientes a diferentes personas o a la misma.

3.4 Revocar Claves

Existen diversos motivos por los que se puede desear revocar una clave. Por ejemplo, si la clave secreta ha sido robada, o si se ha olvidado la contraseña de ésta. En cualquier caso la orden de revocación es:

```
$ gpg --gen-revoke
```

Esto creará un **Certificado de Revocación**. Para ello es necesaria la clave secreta, de lo contrario cualquiera podría hacer un certificado y revocar una clave que no le perteneciera. En el caso anterior en el que la contraseña ha sido olvidada, se hace imposible generar un certificado de revocación. Por este motivo es muy aconsejable generar un certificado de revocación a continuación de la generación de la clave. Es primordial guardar este certificado en un lugar seguro para que nadie pueda usarlo y revocar la clave.

3.5 Administrar las Claves

Existe un fichero que es a modo de una base de datos, en el que se guardan todos los datos relacionados con las claves, incluido los valores relativos al grado de confianza (*Ownertrust*); para más información sobre esto leer 3.6 (Firmar las claves). Con orden

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

se muestran todas las claves presentes. Para poder ver también las firmas en cada clave, usar:

```
$ gpg --list-sigs
```

(ver 3.6 (Firmar las claves) para más información). Para ver las huellas digitales (*fingerprints*):

```
$ gpg --fingerprint
```

Las «huellas digitales» sirven para confirmar la identidad de la persona. Esta orden nos muestra una lista alfanumérica que podemos comprobar, por ejemplo, por teléfono.

Para ver el listado de las claves secretas:

```
$ gpg --list-secret-keys
```

Nota: el listado de las huellas digitales y las firmas de las claves secretas no es de ninguna utilidad.

Para eliminar una clave pública:

```
$ gpg --delete-key UID
```

Para eliminar una clave secreta:

```
$ gpg --delete-secret-key
```

Existe otra orden que es relevante para trabajar con las claves:

```
$ gpg --edit-key UID
```

Para esta orden necesitaremos la contraseña, y podemos, entre otras cosas, editar la fecha de caducidad, añadir una huella digital y firmar la clave.

3.6 Firmar las Claves

Como se ha mencionado anteriormente en la introducción, existe un talón de Aquiles en el sistema: la autentificación de las claves públicas. Si se obtiene una clave pública errónea, ya se puede despedir uno del valor del cifrado. Para evitar estos riesgos está la posibilidad de firmar las claves. Cuando tenemos la certeza de que una clave es válida y pertenece a quien dice, podemos firmarla digitalmente, de modo que otros que confíen en nuestra firma la puedan dar por válida.

Usando el comando

```
$ gpg --edit-key UID
```

para la clave que queremos firmar, nos llevará al subcomando

```
Command> sign
```

¡¡¡Sólo se debe firmar una clave cuando se esté ABSOLUTAMENTE SEGURO de que dicha clave es auténtica!!! En realidad, sólo se puede estar totalmente seguro cuando la clave se ha recibido en mano, o por ejemplo si se ha recibido por correo y a continuación se han comprobado las huellas digitales por correo. Una clave no debe ser nunca firmada en base a una suposición.

Basándose en las firmas existentes en una clave y en el «grado de confianza», `GnuPG` determina la validez de las claves. El grado de confianza (*Ownertrust*) es un valor que el propietario de una clave usa para determinar el nivel de confianza para una cierta clave. Estos valores son:

- 1 = Don't know (No sé, no conozco)
- 2 = I do NOT trust (Confianza nula)
- 3 = I trust marginally (Confianza marginal)
- 4 = I trust fully (Confianza plena)

Si el usuario no se fía de una firma puede indicarlo así, y rechazar la confianza en ésta. La información sobre la confianza no se guarda en el mismo fichero que el de las claves, sino en otro diferente.

4 Codificar y Descodificar

Después de haber instalado y configurado todo tal y como queríamos, podemos comenzar a cifrar y descifrar datos.

Es posible que cuando estemos cifrando o descifrando un documento, tengamos más de una clave privada en nuestro anillo de claves privadas. Si esto es así, es necesario seleccionar una de ellas como activa. Para ello se puede usar la opción

```
-u UID
```

o bien la opción

```
--local-user UID
```

También podemos agregar la siguiente línea en el fichero de configuración `$HOME/.gnupg/options`:

```
default-key UID
```

Si se desea indicar el UID de un destinatario para cifrar un fichero con su clave, se puede hacerse con la opción

```
-r
```

o la opción

```
--recipient
```

4.1 Codificar

La orden para cifrar es la siguiente:

```
$ gpg -e [fichero]
```

ó

```
$ gpg --encrypt [fichero]
```

Estas órdenes cifrarían un fichero con la clave que hayamos definido por defecto en el fichero de configuración `options`. Para cifrar un fichero con la clave de otro usuario:

```
$ gpg -er Destinatario [fichero]
```

Pero como ya hemos comentado anteriormente esto produciría un criptograma con el nombre de `fichero.gpg`; se puede añadir la opción

```
--armor
```

para que el criptograma sea del tipo 7-bit ASCII:

```
$ gpg -a -er Destinatario [fichero]
```

que producirá un criptograma con la extensión `fichero.asc`. Ya que los mensajes, ficheros, y otro tipo de datos que enviamos codificados van cifrdos con la clave del destinatario, existe el riesgo de que alguien lo haga suplantando nuestra identidad. Para evitar esto basta con firmar digitalmente todo lo que se cifre (ver 5.2 (Firmas)).

4.2 Descodificar

El comando para descifrar es:

```
$ gpg [-d] [fichero]
```

ó

```
$ gpg [--decrypt] [fichero]
```

En este caso no es necesaria la opción, es opcional, ya que la orden `gpg` usa por defecto la opción

```
--decrypt.
```

En todos los casos que hemos nombrado aquí el resultado está direccionado a `stdout`, pero puede ser redireccionado con la opción

```
-o [fichero]
```

a un fichero con cualquier otro nombre.

5 GnuPG + PGP

Al ser `PGP` un programa más antiguo que `GnuPG`, es normal que un nuevo usuario de `GnuPG` tenga ya instalado alguna versión de `PGP` en su sistema, y que desee mantener sus viejas claves después de actualizarse a `GnuPG`. Pues bien, no sólo es posible importar el contenido de los anillos de claves sino que, alternativamente, es posible que `GnuPG` gestione los anillos de claves de `PGP 2.6.3` y `PGP 5.0`.

Hay otros «problemas» de compatibilidad sobre los que también trataremos en este capítulo, como son las firmas de tipo **V4** generadas por `GnuPG`, o el uso por parte de `PGP` de los algoritmos propietarios `RSA` o`IDEA`. Empezaremos por esto último.

5.1 Uso de Algoritmos *no libres*

El uso de algoritmos con patentes restrictivas por parte de `PGP` representa un problema en tanto en cuanto la filosofía en torno a `GnuPG` es la de implementar un sistema criptográfico libre. Así pues, las patentes sobre estos algoritmos imposibilitan una implementación total. Pero `GnuPG` también pretende cumplir con las reglas de los «estándares» de *OpenPGP* .

Existen unas extensiones para *RSA* e *IDEA* que pueden ser instaladas y que permiten cierto uso de estos algoritmos. Las claves generadas por `PGP 2.6.x` son del tipo `RSA`, y el algoritmo de cifrado que usa es `IDEA` (también puede ser usado por `PGP 5.x`). Es posible conseguir el código fuente de estos algoritmos en los ficheros *rsa.c* e *idea.c* .

También existen los binarios instalables de estas extensiones para algunas distribuciones de *Linux* , como *Debian* (comprobar para otras distribuciones).

5.2 Firma Digital con GnuPG

`GnuPG` es el único sistema capaz de implementar firmas digitales **V4** (de acuerdo con *OpenPGP*) y esta es la opción por defecto, pero en este caso `PGP` no es capaz de verificarlas. Es posible obligar a `GnuPG` a usar **V3**, de dos modos:

- Indicarlo en el fichero de configuración `$HOME/.gnupg/options` añadiendo la línea:

```
force-v3-sigs
```

- Indicar esta opción cada vez que se desee firmar un mensaje en **V3**:

```
$ gpg <opción> --force-v3-sigs [fichero]
```

5.3 Importar Anillos de Claves de PGP a GnuPG

Intentaremos explicar cómo exportar las claves públicas y privadas desde nuestros anillos de claves PGP a los anillos de claves GnuPG.

NOTA: este método se ha extraído del *PGP2GnuPG* Howto de Caskey L. Dickson y no lo he probado personalmente. La última actualización del «PGP2GnuPG Howto» data del data de Diciembre de 1998. Por ello, y para poder integrar PGP con GnuPG, recomiendo el uso del método que se explica en la 5.4 (siguiente sección) por ser más sencillo y fiable.

Suponiendo que tengamos instaladas las dos versiones de PGP para Unix/Linux, tenemos pues sus respectivos anillos de claves públicas y privadas en `$HOME/.pgp/`:

- pubring.pgp -> fichero de claves públicas de `PGP 2.6.x`
- secring.pgp -> fichero de claves privadas de `PGP 2.6.x`
- pubring.pkr -> fichero de claves públicas de `PGP 5.x`
- secring.skr -> fichero de claves privadas de `PGP 5.x`

A continuación usaríamos las órdenes que correspondan a cada versión para extraer la(s) clave(s) que deseemos.

Así, para extraer una clave de `PGP 2.6.x`:

```
$ pgp -kx UID fichero anillo
```

vg.:

```
$ pgp -kx Pepe clavepepe2 ~/.pgp/pubring.pgp
```

Esta operación nos daría el fichero clavepepe2.pgp. Para extraer nuestra clave privada, no tendríamos más que indicar nuestro UID y el fichero de las claves secretas `~/.pgp/secring.pgp`. No me consta que haya modo alguno de indicar más de un UID con `PGP 2.6.3`, si sabéis de alguno, por favor enviadme un *mensaje* .

Una vez extraída la clave sólo queda importarla al fichero de GnuPG:

```
$ gpg --import clavepepe2
```

Para extraer una clave de `PGP 5.0`:

```
$ pgpk -x UID -o fichero
```

vg.:

```
$ pgpk -x Pepe -o clavepepe5
```

En este caso, el fichero por defecto es el de las claves públicas, y obtendríamos el fichero clavepepe5 como hemos indicado.

Una vez más, sólo queda importar la clave:

```
$ gpg --import clavepepe5
```

Ya que PGP 5.0 no nos permite indicarle el fichero sobre el que queremos operar, la extracción de la clave secreta se complica algo. La solución viene dada por un sistema superior como GnuPG:

Este procedimiento pone en riesgo la clave secreta durante un breve periodo de tiempo, así que no debería ser usado en un sistema multiusuario o público. Los pasos a seguir son:

- Extraer la clave pública correspondiente a la clave privada que queremos exportar, e importarla a GnuPG.
- ¡¡¡Borrar la contraseña de la clave secreta!!! (se recomienda hacer una copia de seguridad del fichero secring.skr):

```
$ pgpk -e UID
```

vg.:

```
$ pgpk -e 0x614DB9FA

sec  1024 0x614DB9FA 1998-03-22 ---------- DSS        Sign & Encrypt

sub  1024 0x2B9E0571 1998-03-22 ---------- Diffie-Hellman

uid  Horacio <homega@vlc.servicom.es>

uid  Horacio <homega@correo.com>


1024 bits, Key ID 0x614DB9FA, created 1998-03-22

"Horacio <homega@vlc.servicom.es>"

"Horacio <homega@correo.com>"

Do you want to unset this key as axiomatic [y/N]? N

Do you want to unset this key as axiomatic [y/N]? N

Do you want to add a new user ID [y/N]? N

Do you want to change your pass phrase (y/N)? Y
Need old passphrase. Enter pass phrase: <introducir contraseña>
Need new passphrase. Enter pass phrase: <dejar vacío>
Enter it a second time. Enter pass phrase: <dejar vacío>
Changing master key passphrase...

Changing subkey passphrase...

Do want to set this as your default key [y/N]? N

Keyrings updated.
```

- El paso siguiente será exportar la clave privada. Como ya hemos podido ver, PGP 5.0 es incapaz de hacer esto, así que usaremos GnuPG:

```
$ gpg --export-secret-keys --secret-key-ring ~/.pgp/secring.skr 0x614DB9FA > miclave
```

Todo esto en una una sola línea, y se creará el fichero `miclave`.

- Ahora ya podemos importar la clave secreta a GnuPG:

```
$ gpg --import < miclave
```

Acto seguido volveremos a introducir una contraseña a la clave desde `GnuPG`.

5.4 Usar Anillos de Claves de PGP con GnuPG

Es posible evitar todo lo anterior, manteniendo instaladas las diferentes versiones de `PGP` al mismo tiempo que la de `GnuPG`. Siendo `GnuPG` un sistema superior y más nuevo, puede reconocer los anillos de claves de `PGP` como propios.

En el caso de `PGP 5.0`, basta con añadir el camino completo a los ficheros de claves de `PGP 5.0`, precedido por `keyring` o `secret-keyring`, al final del fichero `~/.gnupg/options` según corresponda:

```
keyring ~/.pgp/pubring.pkr
secret-keyring ~/.pgp/secring.skr
```

Los ficheros de claves de `PGP 2.6.3` son reconocidos por `GnuPG` por defecto. Si esto no fuera así, bastaría con repetir la misma operación anterior adaptándola a las circunstancias:

```
keyring ~/.pgp/pubring.pgp
secret-keyring ~/.pgp/secring.pgp
```

Si a continuación hacemos un listado de las claves públicas con `GnuPG`, observaremos que lee los tres ficheros, `~/.gnupg/pubring.gpg`, `~/.pgp/pubring.pkr`, y `~/pubring.pgp`:

```
$ gpg --list-keys

     /home/usuario/.gnupg/pubring.gpg
     --------------------------------
     pub 1024D/57548DCD 1998-07-07 Werner Koch (gnupg sig)
     <dd9jn@gnu.org>
     pub 1024D/A95AF46C 1998-11-29 Brenno J.S.A.A.F. de Winter
     <brenno@dewinter.com>
     sub 3072g/A3CA62A0 1998-11-29

   (... y demás claves públicas DSA/ElGamal...)

     /home/usuario/.pgp/pubring.pkr
     ------------------------------
     pub 1024D/FAEBD5FC 1997-04-07 Philip R. Zimmermann <prz@pgp.com>
     sub 2048g/42F0A0A0 1997-04-07

   (... etc DSS/Diffie-Helman...)

     /home/usuario/.pgp/pubring.pgp
     ------------------------------
     pub 1024R/88A17FF5 1995-09-11 IRIS-CERT, Spain

   (... etc RSA...)
```

Lo mismo sucedería con las claves privadas:

```
$ gpg --list-secret-keys

     /home/horacio/.gnupg/secring.gpg
     --------------------------------
     sec 1024D/42337AE6 1999-03-14 Horacio (comentario)
     <homega@vlc.servicom.es>
     ssb 2048g/1F177864 1999-03-14

     /home/horacio/.pgp/secring.skr
     ------------------------------
     sec 1024D/7992AB40 1998-05-04 Horacio <homega@vlc.servicom.es>
     uid Horacio <homega@correo.com>
     ssb 2048g/917366AE 1998-05-04

     /home/horacio/.pgp/secring.pgp
     ------------------------------
     sec 1024R/32D4A925 1997-09-23 Horacio <homega@vlc.servicom.es>
```

6 Firmar y Verificar

Firmar y verificar firmas es una parte importante de los sistemas de criptografía de clave pública. El usuario puede firmar una serie de datos o un documento en varios modos, para lo que usa su propia clave privada. Para verificar las firmas de otros usuarios, es necesario poseer previamente las claves públicas de éstos.

6.1 Firmar

Para firmar un fichero con la clave propia se usa la orden

```
$ gpg -s [fichero]
```

ó

```
$ gpg --sign [fichero]
```

Esta orden, además de producir una firma digital, también comprime el fichero, por lo que el resultado es un fichero de tipo binario (y por tanto ilegible). Para producir un fichero firmado legible (ascii), se usa la orden

```
$ gpg --clearsign [fichero]
```

De este modo, tanto la firma como los datos firmados, son legibles con un editor.

Cuando queramos que la firma aparezca en un fichero separado, sobre todo cuando se trata de firmar un fichero binario, como por ejemplo un archivo comprimido, o un ejecutable, usaremos la orden

```
$ gpg -b [fichero]
```

ó

```
$ gpg --detach-sign [fichero]
```

Este es el modo que MIME/PGP usa para firmar los mensajes del correo electrónico. Este modo es muy útil cuando tengamos que firmar un binario, por ejemplo, para distribuirlo, ya que la firma se basa en el binario pero va en un fichero aparte. La opción `-armor` también puede ser de utilidad en estos casos.

A menudo debemos cifrar y firmar un fichero a un tiempo. La orden que usaríamos en este caso sería

```
$ gpg [-u Remitente] [-r Destinatario] [--armor] --sign --encrypt [fichero]
```

La funcionalidad de las opciones `-u (-local-user)` y `-r (-recipient)` es la que se ha descrito ya anteriormente.

6.2 Verificar

Al descifrar un criptograma que también haya sido firmado digitalmente, la firma es verificada automáticamente. En todo caso es posible verificar la firma simplemente con la orden

```
$ gpg [--verify] [fichero]
```

7 Fuentes de Información

7.1 GnuPG

- *Página Principal de GnuPG* (en inglés)
- *Archivos de la lista de correo de GnuPG* (en inglés)
- La información contenida en el paquete de instalación y/o de fuentes, sobre todo:

```
$ gpg --help
```

7.2 PGP

`PGP` es el programa de cripto más antiguo y, de momento, más extendido. Se ha escrito mucha documentación en torno a `PGP`. Mucha de esta información se puede usar para entender mejor el funcionamiento de `GnuPG`.

- *Página Internacional de PGP* . Desde aquí es posible acceder a mucha información sobre `PGP` en varias lenguas (aunque principalmente en inglés).
- *PGP DH vs. RSA FAQ* . Preguntas y respuestas sobre las diferencias entre los algoritmos "Diffie-Hellman" y "RSA".

7.3 Recursos en Castellano

Existen multitud de recursos en la red para hispanohablantes. Aquí nombraremos un par de ellos desde los que podremos acceder a muchos otros:

- Página del boletín electrónico sobre seguridad *Kriptópolis* .
- Páginas sobre PGP de *RedIris*

7.4 Servidores de Claves

- *Keyserver Net*
- *http://wwwkeys.eu.pgp.net/*
- *Servidor de RedIris*

7.5 Libros

- B. Schneier, “Applied Cryptography, Second Edition”, Wiley, 1996
  Deutsche Ausgabe unter dem Titel “Angewandte Kryptographie”, Addison-Wesley, 1996

8 Sobre este Documento

8.1 Versiones

Versión original en alemán: La **versión 0.1** fue la primera versión en alemán.

Todos los cambios para la versión original, en alemán, están documentados en un fichero diff: *dieses Dokument*

- **English version 0.1.0**, del 30 de Mayo de 1999. Esta versión es una traducción de la versión alemana al inglés, con algunos cambios.
- **Deutsche Version 0.1.1**
  - Nueva sección «Límites en seguridad»
  - Mejorada la explicación sobre firmas
  - Varios cambios sugeridos por Werner Koch (¡gracias!)
- **Versión 0.1.2**, del 29 de Mayo de 1999 (Anno MMDCCLII ad Urbe condita). Esta versión en castellano es una traducción de la versión inglesa, y se han hecho algunos cambios. Se ha añadido el capítulo 5 sobre compatibilidad e interoperabilidad de `GnuPG` con `PGP`.

- **Versión 0.1.3**, del 28 de Septiembre de 1999. Reescrito a código SGML (LinuxDoc) desde el código HTML. Corrección de algunos errores en castellano.

Notas para la versión española: Cualquier comentario o corrección al documento que ayude a mejorarlo es bienvenido. Por favor, enviad cualquier sugerencia a <*homega@ciberia.es*> .

Notas para la versión inglesa: All remarks for this document can be send to Brenno J.S.A.A.F. de Winter <*brenno@dewinter.com*> . Comments help us make a better document and are greatly appreciated.

Notas para la versión alemana: Anregungen, Kritik, Verbesserungen und Erweiterungen einfach an Michael Fischer v. Mollard <*fischer@math.uni-goettingen.de*> senden, damit dieses Dokument weiter verbessert werden kann.