Substituir PGP 2.x por GnuPG

Este documento está basado en una guía de compatibilidad PGP 2.x/GnuPG (http://www.toehold.com/~kyle/pgp-compat.html) anterior, desarrollada por Kyle Hasselbacher (<kyle@toehold.com>). La guía fue reeditada y ampliada por Mike Ashley (<jashley@acm.org>). Michael Fischer v. Mollard (<mfvm@gmx.de>) transformó el código HTML a DocBook SGML, y también añadió algunos datos. Algunos de los temas aquí tratados tienen su origen en las listas de correo de `gnupg-devel` y `gnupg-user`. La solución dada al problema de firmar y cifrar a un tiempo con una clave RSA fue tomada del guión de compatibilidad (http://muppet.faveve.uni-stuttgart.de/~gero/gpg-2comp/changes.html) de Gero Treuner. Para cualquier duda, error, o sugerencia sobre este manual, diríjase al mantenedor de este documento, Mike Ashley (<jashley@acm.org>). Para cualquier duda, corrección, o sugerencia sobre la versión en castellano, diríjase al traductor, Horacio (<homega@ciberia.es>).

1. Introducción

Este documento describe cómo comunicarse con otras personas que todavía estén usando viejas versiones de PGP 2.x. GnuPG puede usarse como un substituto completo de PGP 2.x. Con GnuPG es posible cifrar y descifrar mensajes PGP 2.x, importando primero las claves viejas, pero no se pueden generar claves de PGP 2.x. En este documento se demuestra cómo ampliar la distribución normal de GnuPG para que funcione con claves PGP 2.x, y muestra qué opciones deben ser usadas para asegurar la interoperabilidad con los usuarios de PGP 2.x. También se avisa de anomalías en la interoperabilidad entre PGP 2.x y GnuPG.

Nota: El uso de los módulos de extensión `idea.c` y `rsa.c` sin las correspondientes licencias de estos algoritmos puede ser ilegal. En este documento no se recomienda que se usen dichos módulos. Si Vd. dispone de claves PGP 2.x, el autor sugiere que las revoque en favor de otras nuevas y que anime a las personas con las que mantenga correspondencia y que continúen usando claves PGP 2.x, a que hagan lo mismo.

2. Ampliar GnuPG para su funcionamiento con claves PGP 2.x

La distribución normal de GnuPG no funciona con claves PGP 2.x debido a que PGP 2.x hace uso de IDEA como algoritmo de cifrado simétrico, y de RSA como algoritmo de clave pública. Estos dos algoritmos están patentados[1]y sólo pueden ser usados bajo ciertas condiciones restrictivas. La política de

GNU es la de no hacer uso de algoritmos patentados, dado que éstas son una contradicción con el espíritu del "software" libre. La utilización de estos algoritmos representa una barrera para el uso libre de GnuPG.

El uso de RSA e IDEA sin una licencia sobre éstos puede ser o no legal dependiendo de varias cuestiones. RSA sólo está patentado en los Estados Unidos, por lo tanto sí que es lícito desarrollar versiones de RSA fuera de los EE.UU. El módulo de extensión de RSA para GnuPG es una de estas versiones, y en consecuencia sí que puede ser usado legalmente fuera de los Estados Unidos, aunque sería ilegal si lo usara dentro de este país. En los Estados Unidos existe una implementación de referencia para RSA, llamada RSAREF, y que se encuentra disponible por ftp en funet.fi (ftp://ftp.funet.fi/pub/crypt/cryptography/asymmetric/rsa/rsaref2.tar.gz), o en debian.org (ftp://non-us.debian.org/debian-non-US/dists/stable/non-US/source/rsaref_19930105.orig.tar.gz), y que puede usarse legalmente en los EE.UU. sin cargo alguno para el uso con fines no lucrativos. Debido a leyes que restringen la exportación de este código de los EE.UU., no puede ser distribuido fuera de este país, y por tanto existen dos modos de integrar RSA en GnuPG: uno para los EE.UU. y Canadá, y otro para el resto del mundo.

La situación de IDEA es más simple. IDEA está patentado en Europa y en los EE.UU., y queda pendiente una patente para Japón. El propietario de la patente, Ascom, concede una licencia (http://www.ascom.ch/infosec/idea/licensing.html) con fines no lucrativos gratuita, pero la definición de fines no lucrativos es bastante estricta. Si desea utilizar IDEA para fines comerciales necesita adquirir una licencia.

Para poder usar los módulos de extensión primero hay que obtener el código fuente de éstos, `idea.c` y `rsa.c`, o `rsaref.c`, del directorio de cotribución (ftp://ftp.gnupg.org/pub/gcrypt/contrib/) de código a GnuPG. Una vez se tenga el código, éste debe ser compilado. Si se usa gcc, la compilación será como sigue:

```
alice% gcc -Wall -O2 -shared -fPIC -o idea idea.c
[...]
alice% gcc -Wall -O2 -shared -fPIC -o rsa rsa.c
[...] # ó
alice% gcc -Wall -O2 -shared -fPIC -o rsa rsaref.c /usr/lib/rsaref.a
```

El último argumento `/usr/lib/rsaref.a` se debe substituir con el camino real de la biblioteca RSAREF en el sistema.

Una vez compilado, GnuPG debe recibir las instrucciones para cargarlos. Esto se puede hacer usando la opción `load-extension`, bien desde la línea de órdenes, o bien desde el fichero de opciones, aunque por regla general se hará desde el fichero de opciones. Por ejemplo, si se ha puesto los binarios compilados `idea` y `rsa` en el directorio `~/.gnupg`, en el fichero de opciones se debe añadir

```
load-extension ~/.gnupg/idea
load-extension ~/.gnupg/rsa
```

Si no se especifica un camino de modo explícito, GnuPG busca los módulos de extensión en el directorio de módulos de GnuPG por definición, el cual es `/usr/local/lib/gnupg`. Si se ha compilado GnuPG con prefijo distinto para el directorio de instalación, usando `--prefix PREFIX` durante la configuración del código fuente de GnuPG, entonces el directorio de módulos será `PREFIX/lib/gnupg`. En tal caso, copiar los dos ficheros ‘rsa’ e ‘idea’ en el directorio de módulos descrito arriba. Asegúrese de que estos ficheros tienen los permisos correctos. No es necesario hacer los ficheros ejecutables, ya que estos ficheros no son programas sino módulos compartidos, y por tanto deben tener permiso de lectura para todos.

3. Importar claves PGP 2.x

Una vez que las extensiones han sido cargadas, el importar un par de claves de PGP 2.x es una tarea fácil usando la opción `import`. Aun así se debe tener cuidado con un par de detalles.

- No se debe exportar una clave privada desde PGP 2.x en forma de fichero en armadura ASCII. Ya que PGP 2.x es anterior a la especificación OpenPGP, la cabecera del mensaje en armadura que usa PGP 2.x no es conforme con OpenPGP. Debido a que la exportación de una clave privada es un caso poco común, GnuPG no comprueba que el fichero en armadura ASCII sea una clave privada.
- GnuPG presupone que las claves públicas importadas van autofirmadas por sus correspondientes claves públicas. Esta es una precaución bastante prudente, y tanto GnuPG como las nuevas versiones de PGP autofirman las claves públicas durante el proceso de su generación. Sin embargo, PGP 2.x no lo hace. Para solventarlo, se puede autofirmar la clave pública antes de exportarla desde PGP 2.x. De modo alternativo, se puede usar la opción `allow-non-selfsigned-uid` para forzar a GnuPG a aceptar la clave. Se recomienda que la clave sea autofirmada antes de ser exportada, o incluso después de haber sido importada usando la opción anterior, ya que el uso de una clave no autofirmada es un riesgo para la seguridad.

```
alice% pgp -kx alice public.pgp
Pretty Good Privacy(tm) 2.6.2 - Public-key encryption for the masses.
[...]
Extracting from key ring: '/u/alice/.pgp/pubring.pgp', userid "alice".
Key for user ID: Alice <alice@cyb.org>
1024-bit key, Key ID 24E2C409, created 1999/09/18

Key extracted to file 'public.pgp'.

alice% pgp -kx alice private.pgp .pgp/secring.pgp
Pretty Good Privacy(tm) 2.6.2 - Public-key encryption for the masses.
[...]

Extracting from key ring: '.pgp/secring.pgp', userid "alice".
Key for user ID: Alice <alice@cyb.org>
1024-bit key, Key ID 24E2C409, created 1999/09/18

Key extracted to file 'private.pgp'.

alice% gpg --import public.pgp
gpg: key 24E2C409: public key imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)

alice%gpg --import private.pgp
gpg: key 24E2C409: secret key imported
gpg: Total number processed: 1
gpg:       secret keys read: 1
gpg:   secret keys imported: 1
```

4. Usar claves PGP 2.x

Una clave pública importada se puede usar para cifrar documentos para un usuario de PGP 2.x y para verificar firmas que hayan sido generadas con una clave privada PGP 2.x *Es importante comprender que no es posible usar una nueva clave OpenPGP para comunicarse con un usuario PGP 2.x, por tanto es necesario importar una clave vieja PGP 2.x para esta tarea.*

4.1. Cifrar un documento para un usuario de PGP 2.x

Para cifrar un documento se usan varias opciones en la línea de órdenes, y el documento que se va a cifrar debe ser especificado como un fichero.

```
alice% gpg --rfc1991 --cipher-algo idea --compress-algo 1 --encrypt --recipient alice secret
gpg:
RSA keys are deprecated; please consider creating a new key and use this key in the future
gpg: this cipher algorithm is depreciated; please use a more standard one!
```

Cada una de las opciones en la línea de órdenes son necesarias.

- La opción `rfc1991` se usa para forzar GnuPG a que sea más conforme con RFC 1991, que es la antigua especificación PGP implementada en PGP 2.x. Si se omite esta opción, la salida de GnuPG estará malformada e inutilizable por PGP 2.x.
- La opción `cipher-algo` especifica el algoritmo de cifrado simétrico con el que el documento será cifrado. En el caso especial de cifrar un documento para una clave pública de PGP 2.x, el algoritmo de cifrado que se especifique debe ser IDEA. Si se omitiera esta opción, el documento se cifrará generalmente con 3DES, un algoritmo que no está implementado en PGP 2.x.
- El algoritmo de compresión indica cómo se formará el resto de la orden. La opción `compress-algo` especifica a GnuPG que debe usar el viejo algoritmo de compresión zlib, que es el usado por PGP 2.x. A pesar de esto, GnuPG usa cabeceras de longitud parcial cuando cifra una cadena de tamaño desconocido, y esto no está implementado en PGP 2.x. El documento que se vaya a cifrar debe estar por lo tanto, en un fichero de modo que GnuPG sepa el tamaño total del documento a cifrar antes de comenzar. Por consiguiente, no es posible usar tuberías al utilizar claves PGP 2.x.

4.2. Firmar un documento para un usuario de PGP 2.x

Firmar un documento con una clave vieja no es diferente a hacerlo con una clave nueva.

```
alice% gpg --local-user 0x24E2C409 --sign document
You need a passphrase to unlock the secret key for
user: "Alice <alice@cyb.com>"
1024-bit RSA key, ID 24E2C409, created 1999-09-18

gpg: RSA keys are deprecated; please consider creating a new key and use this
key in the future
```

En este ejemplo, la opción `local-user` se usa para especificar qué clave privada se utilizará para firmar. El fichero de salida es de la forma `document.gpg`. Si la firma va a ser verificada usando PGP 2.x, se debe renombrar a un nombre de fichero con la extensión `.pgp`.

4.3. Firmar y cifrar un documento para un usuario de PGP 2.x

GnuPG no posee una implementación nativa para firmar un documento con una clave RSA y al mismo tiempo cifrarlo con una clave RSA. Sin embargo es posible usar una solución que requiere que se lleven a cabo unos cuantos pasos anteriormente. El proceso implica la creación de una firma acompañante y a continuación el uso de ésta para crear un fichero cifrado que pueda ser descifrado y verificado usando PGP 2.x.

Hay cuatro pasos. El primer paso genera una firma acompañante.

```
alice% gpg --detach-signature --recipient alice --local-user 0x24E2C409 document

You need a passphrase to unlock the secret key for
user: "Alice <alice@cyb.com>"
1024-bit RSA key, ID 24E2C409, created 1999-09-18

gpg: RSA keys are deprecated; please consider creating a new key and use this
key in the future
```

El segundo paso convierte el documento a un formato interno, literal, que va descifrado.

```
alice% gpg --store -z 0 --output document.lit document
```

El tercer paso combina la firma acompañante con el documento literal. Es esto lo que PGP 2.x usa para verificar la firma después de descifrarlo.

```
alice% cat Notes.sig Notes.lit | gpg --no-options --no-literal --store --compress-algo 1 --output document.z
gpg: NOTE: --no-literal is not for normal use!
```

El cuarto y último paso implica el uso de GnuPG para descifrar el texto plano y la firma con el objeto de producir un documento firmado y cifrado que pueda ser descifrado y verificado por PGP 2.x.

```
alice% gpg --rfc1991 --cipher-algo idea --no-literal --encrypt --recipient alice --output document.pgp document.z
gpg: NOTE: --no-literal is not for normal use!
gpg: RSA keys are deprecated; please consider creating a new key and use this
key in the future
gpg: this cipher algorithm is depreciated; please use a more standard one!
```

El documento firmado y cifrado también puede ir en armadura ASCII mediante el uso de las opciones de rigor.

```
alice% gpg --rfc1991 --cipher-algo idea --no-literal --encrypt --recipient alice --output document.asc --armor documen
gpg: NOTE: --no-literal is not for normal use!
gpg: RSA keys are deprecated; please consider creating a new key and use this
key in the future
gpg: this cipher algorithm is depreciated; please use a more standard one!
```

4.4. Descifrar un documento gpg

Una clave privada importada se puede usar para descifrar documentos cifrados para esa clave, así como generar firmas mediante el uso de esa clave. Descifrar un mensaje no es, en este caso, más difícil que cuando se usa cualquier otra clave.

```
alice% gpg secret.pgp

You need a passphrase to unlock the secret key for
```

```
user: "Alice <alice@cyb.org>"
1024-bit RSA key, ID 24E2C409, created 1999-09-18

gpg: NOTE: cipher algorithm 1 not found in preferences
gpg: secret.pgp: unknown suffix
Enter new filename [secret]:
```

De nuevo, en este caso, se puede evitar el aviso de error renombrando el fichero de entrada con una extensión `.gpg`. Si se ve un aviso de GnuPG del tipo “cipher algorithm 1 not found in preferences”, se puede ignorar sin problemas.

4.5. Verificar una firma de PGP 2.x

Verificar una firma usando generada con una clave PGP 2.x es sencillo.

```
alice% gpg document.pgp
gpg: document.pgp: unknown suffix
Enter new filename [document]:
File ‘document’ exists. Overwrite (y/N)? y
gpg: old style (PGP 2.x) signature
gpg: Signature made Sat Sep 18 17:55:30 1999 EST using RSA key ID 24E2C409
gpg: Good signature from "Alice <alice@cyb.org>"
```

El diálogo para renombrar el fichero puede ser evitado si el documento que se quiere verificar ha sido renombrado con una extensión `.gpg` antes de invocar gpg.

5. Trabajar con firmas sobre texto

Desde la aparición de la versión 1.0 de GnuPG, existe un problema con el intercambio de documentos con firma sobre el texto entre GnuPG y todas las versiones de PGP. Parece ser que las dificultades se deben a desviaciones en la implementación de la especificación OpenPGP. Respecto a PGP 2.x, las firmas generadas por éste se pueden verificar usando GnuPG, y las firmas generadas con GnuPG se pueden verificar usando PGP 2.x. El documento resultante del proceso de verificación será, en ambos casos, diferente del documento original. Estas diferencias se limitan a espacios en blanco, y por tanto no deberían afectar la legibilidad de los documentos firmados. En caso de que mantener la completa integridad del documento fuera de importancia, se recomienda evitar el uso de firmas sobre texto. En cualquier caso, se insiste una vez más en la recomendación de usar y recomendar el uso, siempre que sea posible, de las nuevas claves OpenPGP generadas por GnuPG.

Notas

1. La patente de RSA finaliza en Septiembre de 2000. La patente de IDEA finaliza en el año 2011.