

# Privacidad en correos electrónicos con GPG

Daniel E. Coletti  
([dcoletti@xtech.com.ar](mailto:dcoletti@xtech.com.ar))

21 de marzo de 2002

# Capítulo 1

## Introducción

Con el incremento de conexiones hogareñas de banda ancha, la seguridad en Internet, paso a ser un tema aún más importante. Hoy, cualquier persona (sin mucha experiencia) puede llegar a hacer desastres informáticos, utilizando alguna de los cientos de herramientas de *cracking* disponibles en Internet. Con desastres me refieren a ver información “que pasa cerca”, encontrar equipos vulnerables (y vulnerados) rápidamente, entrar a equipos sin permiso, etcétera, etcétera, etcétera.

El servicio más utilizado en Internet (desde sus comienzos) es el correo electrónico, paradójicamente el protocolo que prosperó para transmitir correos, es uno de los más inseguros que existe.

El SMTP (*Simple Mail Transfer Agent*) no hace mucha énfasis en la verificación de la ruta que atravesó el correo, tampoco verifica que la persona que lo envió es quien dice ser, y por supuesto no tiene forma de determinar que el correo que llegó a destino, es el mismo que el que fue enviado originalmente. Para muchas personas esto puede no ser un problema (hoy), pero imagínate un futuro no muy lejano, donde el correo electrónico sea la primordial forma de comunicación, digamos que reemplaza al teléfono, no es que piense que el teléfono es un medio seguro, pero digamos que se requieren “aparatos” para poder hacer maldades con el teléfono y las conversaciones, no sólo software. O imaginemos otras situaciones más actuales donde se utiliza el correo electrónico como una herramienta de trabajo . . . el envío de documentos importantes (cartas, presupuestos, información laboral, hasta un Curriculum Vitae) están a la orden del día. ¿Y si alguien detuvo el correo deliberadamente en el medio del camino? ¿Y si estos documentos fueron cambiados de alguna forma? Las consecuencias te las dejo a tu imaginación . . .

Para comenzar con el proceso de “arreglar” este y otros dilemas concernientes a seguridad de datos enviados, el señor Phil Zimmerman<sup>1</sup> creo la versión 1.0 de PGP – *Pretty Good Privacy*. Originalmente esta versión

---

<sup>1</sup>Ver <http://www.philzimmermann.com> para más información sobre esta persona y la historia de PGP

de PGP era libre, aunque por problemas de patentes y *copyrights* de algunos algoritmos utilizados en PGP, las versiones que le siguieron no lo son. PGP proliferó y llevo a que PGP empezara a ser un estándar en Internet, por lo tanto para el formato de mensajes y claves se creo una RFC (RFC 2440). Para la tranquilidad de los amantes del software libre, los proyectos OpenPGP y GnuPGP comenzaron a calentar motores y hoy existe software que tienen casi las mismas prestaciones que el PGP y estan disponibles para todos. Además yo digo una cosa, no? (si me permitis la interrupción de tu amena lectura), si hay una empresa que controla un algoritmo muy estándar y utilizado de encriptación, ¿qué les impediría poner una pequeña puerta trasera para desencriptar cualquier mensaje? El código es secreto e inaccesible, por lo tanto uno se enteraría de la existencia de esto solamente cuando ya es demasiado tarde. Vaya “poder” para el pais que controle esta empresa, no?<sup>2</sup>.

En este artículo voy a tratar de mostrar el uso de uno de estos softwares, el resultado del proyecto GnuPG: *gpg*, desarrollado primariamente por Werner Koch.

## 1.1. ¿Qué hace el gpg?

El gpg tiene cuatro funciones principales: encriptación de datos, desencriptación de datos, firmar datos criptográficamente, y verificación de firmas digitales. En otras palabras, te permite encriptar archivos, encriptar correos (antes de enviarlos), firmar archivos y/o correos digitalmente, firmar código fuente, validar firmas digitales de otros para determinar si cierto archivo no fue cambiado en su camino y pertenece a quien dice pertenecer. Y otras cosas como mantener anillos de claves (una base de datos de claves) que contienen claves privadas (tus claves) y mantener anillos de claves públicas (las claves de los demás).

### 1.1.1. ¿Para qué te sirve a vos gpg?

Bueno, eso depende . . . para empezar te sirve si quieres que la información que envias pueda ser verificada por quien la recibe, y también te sirve si vos quieres verificar que lo que recibiste esta “entero” y sin cambios. Por esta última razón, yo creo que todas las personas que utilizan código libre necesitan saber utilizar gpg. Se ha tornado algo común, especialmente en código fuente de software de seguridad, firmar digitalmente los archivos, de esta forma todos los que utilizamos estos fuentes podemos estar seguros de que no fue modificado en ninguna parte (como por ejemplo en el servidor ftp del cual descargamos el paquete).

---

<sup>2</sup>Si visitas la página de Zimmermann seguramente vas a encontrarte con mucho de esto

Las otras razones son las más obvias, si quieres enviar correos encriptados y/o firmados digitalmente a tus amigos, parientes, compañeros de trabajo, etc. **que también utilicen gpg o cualquier software compatible con OpenPGP**, necesitas gpg. Si quieres desencriptar las cosas que te mandan tus amigos, parientes, compañeros de trabajo, etc., también necesitas gpg.

## 1.2. ¿Cómo funciona la criptografía de claves públicas?

Acá voy a tratar de ser breve y conciso por dos simples razones, la primera porque yo no soy ningún *guru* de criptografía de claves públicas (por lo tanto aunque quisiera explayarme en el tema no podría ;-)) y la otra porque lo corto y conciso es más simple de entender que lo largo y tedioso.

Cuando se trabaja con este tipo de criptografía hay que tener en cuenta lo siguiente: todo el mundo tiene dos claves, una clave privada y una clave pública.

**La clave pública** se utiliza para **encriptar** datos del dueño de la clave (que van a ser enviados a esta persona) y para verificar algo firmado digitalmente.

**La clave privada** se utiliza para **desencriptar** datos utilizando la clave pública de quien encriptó estos datos originalmente y para crear firmas digitales.

De esto se desprende lo obvio, la clave pública debe estar disponible para todo aquel que la requiera y debe poder viajar libremente por Internet. Por otro lado, la clave privada debe ser guardada celosamente por su dueño y no enviarla a través de ningún transporte inseguro (bajo ningún punto de vista). Solamente vos tenes que tener la posibilidad de desencriptar datos dirigidos a vos únicamente, y vos debes ser la única persona que pueda firmar algo que los demás vean como tuyo.

Lo importante de la clave pública es su integridad como tal (que este “entera” y sin modificación alguna al momento de llegar a destino), no lo secreta o no que es. Hay que tener en claro que la debilidad de este sistema se encuentra en este tema, si nadie puede distinguir entre tu verdadera clave pública y una clave pública modificada (que supuestamente es tuya), tenes un verdadero problema<sup>3</sup>.

Todas las aplicaciones que trabajan con este tipo de criptografía tienen este problema y varias de estas aplicaciones han creado formas de evitar esta debilidad o minimizarla.

GnuPG y PGP tienen mecanismos para validar estas claves, pero depende de

---

<sup>3</sup>Dije que iba a ser breve, por lo tanto si quieres saber más sobre esta debilidad visita <http://www.gnupg.org>

vos si los utilizas o no. Estos mecanismos son utilizando lo que se llamó “*The Web of Trust*” (La Red<sup>4</sup> de Confianza).

### 1.3. La Red de Confianza

La Red de Confianza esta basada justamente en esto “confianza”, la idea es la siguiente . . .

Pedro conoce a Juan y Juan le envía su clave pública, cuando Pedro la recibe llama por telefono a Juan y le pide que le lea alguna parte de la clave que le envió o lo que se llama el *fingerprint*<sup>5</sup>. Cuando todo esta correcto y telefónicamente se verifica que la clave recibida es la correcta, Pedro firma la clave pública de Juan con su clave privada (y la envía a los repositorios de claves). Virna recibe un correo firmado por Pedro, pero en realidad no conoce a Pedro, por lo tanto no podrá confiar plenamente en la clave pública recibida, entonces al verificar la clave ve que Juan firmó la clave pública de Pedro y ella tiene la clave pública de Juan (en la que sí confía), por lo tanto puede confiar en la clave de Pedro. Paso siguiente, Virna firma la clave pública de Pedro y de esa manera se agranda la red de confianza. Esto último sirve para que cuando yo reciba un correo firmado por el desconocido Juan, cuya clave fue firmada por otro desconocido (Pedro), yo pueda confiar en estas dos personas ya que tengo la clave pública de Virna, en la cual sí conozco y confío.

De esta forma se arma La Red de Confianza, y como dije anteriormente, queda en cada uno de nosotros utilizarla y seguir agrandandola.

---

<sup>4</sup>¿Hay alguna traducción seria de “Web”?

<sup>5</sup>Ya describiré esto más adelante

## Capítulo 2

# Instalación de gpg

En este capítulo voy a describir como instalar el software en sus formas más comunes, bajando los fuentes y compilandolos, o utilizando paquetes binarios como RPMs y DEB.

### 2.1. Desde los fuentes

El gpg ya pasó a ser un paquete estándar en la mayoría de las grandes distribuciones, pero nunca esta de más saber como instalar un paquete compilando los fuentes.

Para hacer esto primero hay que bajar los fuentes del <http://www.gnupg.org>, siempre se recomienda mantenerse bien al día con las últimas versiones estables de los paquetes de seguridad, por lo que no viene mal, de vez en cuando, visitar el sitio para tener este paquete al día.

Una vez que se bajaron los fuentes, hay que ejecutar los siguientes comandos (como root):

```
# cd /usr/src
# tar xzvf /tmp/gnupg-<version>.tar.gz
# cd gnupg-<version>
# ./configure
# make
# make check
# make install
```

Dado que las versiones de gnupg van cambiando, cuando ejecutes estos comandos tenes que tener en cuenta de reemplazar `<version>` por la verdadera versión del paquete (por ejemplo “-1.0.9”). Una vez que hayas compilado e instalado todo, el binario **gpg** va a quedar en `/usr/bin/gpg`. Si tenes problemas compilando o instalando el paquete fijate de leer el archivo `INSTALL` y `README` que seguramente te van a ayudar a resolver cualquier problema.

## 2.2. Paquetes binarios

Claramente es mucho mejor, para el usuario común (o cansado de problemas de compilación), instalar el paquete desde su forma binaria. Esto trae un sin número de beneficios (verificación de integridad de los archivos de paquete instalado con RPM por ejemplo). Por lo tanto, esta, es la forma de instalación recomendada.

### 2.2.1. Con RPM

Este formato de paquete es utilizado por muchas distribuciones, entre ellas, RedHat, SuSE, Caldera, Mandrake, Conectiva. En muchas ocasiones no es imprescindible instalar un paquete rpm generado para una distribución en particular en esta distribución (por ejemplo: Un paquete generado para SuSE en un SuSE), pero en la mayoría de las ocasiones esto es lo recomendado. Por lo tanto, dependiendo la distribución que vos utilices te tendrás que dirigir al sitio web de tu distribución (si es que no viene incluida en el CDROM del cual instalaste el Linux) y bajarte el paquete correspondiente. Aunque tengas el paquete en tu CD de instalación, igual te recomiendo que le pegues una visita al sitio web de la distribución ya que puede haber nuevas versiones del gpg.

Para instalar un paquete rpm hay que ejecutar el siguiente comando (como root):

```
# rpm -Uvh /tmp/gpg-<version+plataforma>.rpm
```

Si existe alguna dependencia sobre este paquete te recomiendo que instales el paquete del cual depende el gpg (no utilices `--force` `--nodeps`).

### 2.2.2. Con DEB

Las distribuciones que utilizan este tipo de paquetes son menos, de hecho, las únicas que conozco son Debian y Corel Linux, esta última ya no existe más desde que Microsoft compro una parte de Corel, por lo que digamos que solo sirve en Debian.

Con Debian hay dos formas de instalar paquetes, con “apt-get” o con “dpkg”.

**Con apt-get** se ejecuta:

```
# apt-get install gpg, este comando se encargará de descargar de internet (o file system local) todos los paquetes de los cuales depende gpg
```

**Con dpkg** se requiere tener el archivo `gpg-version.deb` y ejecutar:

```
# dpkg -i gpg-<version>.deb
```

En todos los casos el binario **gpg** queda instalado en `/usr/bin` o `/usr/sbin`.

## Capítulo 3

# Cómo se usa el gpg

En este capítulo voy a describir algunas de las funcionalidades de **gpg**. Básicamente como crear una clave, verificar claves de otras personas, encriptar y descencriptar archivos y también como utilizar los repositorios públicos de claves para que otros tengan la posibilidad de conocer mis claves.

### 3.1. Creando la clave

Una clave de gpg la vamos a necesitar en todos los casos, excepto en la validación de otras claves, con lo cual, si lo que nos interesa realmente es verificar datos firmados digitalmente por otros (como un pedazo de software), no es necesario crear una clave. Pero bueh!, yo lo primero que pensé cuando lei sobre gpg fue ¿Cómo creo mi clave?, ¿Cómo creo mi clave? ¿Cómo creo mi clave?, en fin ... ese soy yo.

El comando para crear una clave de gpg es el siguiente:

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

Un ejemplo de lo que la opción `--gen-key` pregunta es el siguiente:

```
[danx@dorita:danx] gpg --gen-key
gpg (GnuPG) 1.0.6; Copyright (C) 2001 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

gpg: /home/danx/.gnupg: directory created
gpg: /home/danx/.gnupg/options: new options file created
gpg: you have to start GnuPG again, so it can read the new options file
[danx@dorita:danx] gpg --gen-key
```

### *CAPÍTULO 3. CÓMO SE USA EL GPG*

---

gpg (GnuPG) 1.0.6; Copyright (C) 2001 Free Software Foundation, Inc.  
This program comes with ABSOLUTELY NO WARRANTY.  
This is free software, and you are welcome to redistribute it  
under certain conditions. See the file COPYING for details.

```
gpg: /home/danx/.gnupg/secring.gpg: keyring created
gpg: /home/danx/.gnupg/pubring.gpg: keyring created
```

Please select what kind of key you want:

- (1) DSA and ElGamal (default)
- (2) DSA (sign only)
- (4) ElGamal (sign and encrypt)

Your selection? 1

DSA keypair will have 1024 bits.

About to generate a new ELG-E keypair.

    minimum keysize is 768 bits

    default keysize is 1024 bits

    highest suggested keysize is 2048 bits

What keysize do you want? (1024) 1024

Requested keysize is 1024 bits

Please specify how long the key should be valid.

    0 = key does not expire

    <n> = key expires in n days

    <n>w = key expires in n weeks

    <n>m = key expires in n months

    <n>y = key expires in n years

Key is valid for? (0) 1y

Key expires at Mon Feb 3 17:37:09 2003 ART

Is this correct (y/n)? y

You need a User-ID to identify your key; the software constructs the user id  
from Real Name, Comment and Email Address in this form:

"Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: Daniel E. Coletti

Email address: dcoletti@cafelug.org.ar

Comment: CaFeLUG - Argentina

You selected this USER-ID:

"Daniel E. Coletti (CaFeLUG - Argentina) <dcoletti@cafelug.org.ar>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? 0

You need a Passphrase to protect your secret key.

Enter passphrase:

Repeat passphrase:



vencimiento coherente para evitar un potencial *crackeo*<sup>1</sup> de la clave. Yo particularmente (aunque en el ejemplo use 1 año) soy de poner “sin vencimiento”, si alguien *crackea* mi clave en el futuro, ejecutaré el proceso de revocación y listo.

**El Nombre Real** es simplemente el nombre del dueño de la clave

**La Dirección de correo** este dato es sumamente importante y debe escribirse correctamente, especialmente si se va a utilizar programas que soporten gpg como Evolution o Kmail.

**El Comentario** es algo que querramos agregar a la clave que sume para la correcta identificación del dueño de la clave.

Finalmente nos pregunta si esta todo correcto y la frase que sella la clave que estamos generando, este frase es importante y se recomienda fuertemente que se utilice una frase (que contenga espacios). Por supuesto, esto no debe ser algo sumamente extenso, ya que esta frase la vamos a tener que escribir una y otra vez, en cada operación de encriptación, descencipción y al firmar digitalmente algo.

### 3.1.1. Los archivos físicos

Cuando se genera un “par de claves” (acordate que una clave esta integrada por dos en realidad, la pública y la privada) terminan llendo a parar a archivos separados. Todo esto se graba por defecto en el directorio **.gnupg** dentro del *home directory* del usuario.

La clave pública queda en el archivo **/.gnupg/pubring.gpg** y la clave privada en el archivo **/.gnupg/secring.gpg**. Estos archivos deben tener los permisos correspondientes, los permisos que le asigna el gpg por defecto están bien, pero por las dudas aclaro que la clave privada debería tener **400** (solo lectura para el dueño) y la clave pública **644** (lectura/escritura para el dueño y solo lectura para el resto).

## 3.2. El certificado de revocación

Una vez que se crea una clave, se recomienda crear el certificado de revocación, este certificado nos permitirá desactivar una clave en el momento que se requiera, especialmente cuando por alguna razón no podemos recordar la frase que ingresamos al generar la clave privada.

El certificado hay que guardarlo en lugar seguro, al igual que la clave privada, si alguien se hace de este certificado nos puede pasar lo mismo que si alguien nos “roba” la clave privada (archivo *secring.gpg*), por lo tanto

---

<sup>1</sup>Como se inventan palabras cuando uno habla/escribe sobre tecnología, no?

deberá tener los permisos lógicos (400), es más, este certificado se podrá incluso imprimir y guarda en lugar seguro. A diferencia de la clave privada, este certificado se utilizará solo una vez (cuando necesitemos revocar una clave), por lo que se puede poner en un diskette, CDRom o similar y eliminarlo del disco sin problemas.

Para generar este certificado se tiene que ejecutar el siguiente comando:

```
gpg --output cert_revoc_arch.asc --gen-revoke <nombre_de_clave>
```

el archivo “cert\_revoc\_arch.asc” es el nombre del archivo que contendrá el certificado, mientras que **nombre\_de\_clave** es el código identificador que nos asigna el gpg a nuestra clave (algo así como **A1D7637C**<sup>2</sup>).

### 3.3. Preparando las cosas para armar la red de confianza

Cuando ya generamos una clave, y el certificado de revocación esta también creado, podemos empezar a jugar un poco y todo comienza con el firmado de clave propias y ajenas (sin firmar las claves públicas de los demás no vamos a poder encriptar datos para estas personas).

Con lo cual lo primero que vamos a necesitar hacer es enviar nuestra clave pública a alguien. Esto se puede hacer de varias formas, podemos tomar el archivo “pubring.gpg” y enviarlo por correo electrónico (o algun otro medio), aunque hay que tener en cuenta una cosa antes de enviar por correo electrónico nuestra clave pública, en algunos casos puede ser necesario enviar esta clave en texto plano y no en formato “binario” como se encuentra el archivo “pubring.gpg”. Para hacer exportar nuestra clave pública (o cualquier otra que tengamos en nuestro base de datos de claves públicas) se puede ejecutar el siguiente comando:

```
gpg --armor --export
```

La salida de este comando será un choclo de caracteres ASCII algo incomprendible, pero lo importante de esto es que en ASCII y no formato binario. Si queremos enviar la salida a un archivo podemos utilizar el símbolo mayor o la opción `--output <archivo>` del gpg. Por ejemplo

```
gpg --armor --export --output dcoletti.asc
```

---

<sup>2</sup>Este código identificado se puede obtener haciendo “gpg -list-sig nombre\_de\_clave”

Este archivo deberíamos enviárselo a quien sea que necesitemos para que lo firme.

Cuando nuestro amigo recibe este comando va a ejecutar los siguientes comando (**luego de hablar con nosotros por telefono y verificar que la clave que le enviamos esta “entera”**):

```
gpg --import ./dcoletti.asc
```

Para verificar que la clave pública que recibimos es la correcta se pueden hacer algunas cosas más, además de hablar por telefono con el dueño, si la persona que nos envió la clave suele publicar correos en internet (listas de correo, *news groups*, etc.) podríamos ver alguno de estos correos publicado y ver si en la firma de cada correo esta su código de identificación de la clave o su *fingerprint*. Estos dos datos son útiles para poder identificar una clave pública rapidamente. El *fingerprint* es una serie de dígitos hexadecimales que esta pensados para decirlos por telefono rápidamente (son 8 grupos de 4 dígitos hexa), el código de la clave pública es un número mucho más chico que identifica una clave.

Si encontramos correos del dueño de esta clave publicados en Internet (digamos que dos o tres en diferentes sitios) podriamos llegar a pensar que estamos seguros que la clave que recibimos es la clave correcta. Imaginate que quien sea que haya modificado la clave pública que recibiste debería también crackear todos los sitios donde el dueño publicó su *fingerprint* o código de clave y modificar todos estos correos, suena medio loco que alguien tenga la capacidad (y el tiempo) de hacer esto, no?

Bueno, volviendo a la firma de una clave pública ajena . . . Una vez que nuestro amigo recibió la clave, la verificó y la importó, ya esta todo listo para que la firme.

El comando que va a ejecutar es el siguiente:

```
gpg --edit dcoletti@cafelug.org.ar
```

Al ejecutar este comando, nuestro amigo entra en un *subshell* que le permite hacer varias cosas, entre ellas, firmar claves. Fijate que uno de los argumentos que nuestro amigo le pasó al gpg fue mi dirección de correo, dado que lo que le enviamos fue **mi** clave pública él debe importar y luego editar las opciones de **mi** clave pública.

Para firmar la clave lo que él debe ejecutar en este *subshell* es **sign** (firmar en inglés), y luego **save** (guardar en inglés).

La alternativa para hacer esto es (luego de importar la clave, por supuesto) ejecutar:

```
gpg --sign-key dcoletti@cafelug.org.ar
```

En cualquiera de las dos formas, el gpg nos preguntará nuestra frase. Si la frase es correcta firmará la clave pública.

Una vez realizado estas acciones nuestro amigo va a poder encriptar datos y enviarnos a mí (dado que la clave pública que recibió es la mía).

Él deberá hacer lo mismo que hice yo y seguramente yo le devolveré el favor firmando su clave pública.

### 3.4. Encriptado y Desencriptando información

Para encriptar y desencriptar información es necesario tener la clave pública de la persona a la cual le enviaremos esta información encriptada.

El comando para encriptar un archivo es el siguiente:

```
gpg --output shadow.gpg --encrypt --recipient dcoletti@cafelug.org.ar /etc/shadow
```

Cuando yo recibo este archivo por correo (por la forma que fue generado el archivo encriptado seguramente lo habré recibido como un archivo adjunto, y no en el cuerpo del correo, dado que no se utilizó la opción `--armor`), voy a proceder a desencriptarlo utilizando el siguiente comando:

```
gpg --output /home/dcoletti/shadow --decrypt shadow.gpg
```

### 3.5. Firmando y Validando información

Si lo que se necesita es estar seguro que la información llega “entera” y sin modificaciones, uno puede optar por firmar un pedazo de información, en vez de encriptarlo totalmente.

Para hacer esto se puede ejecutar el siguiente comando:

```
gpg --output presupuesto-1002_firm.txt --clearsign presupuesto-1002.txt
```

Cuando el archivo llega a destino, la persona que lo recibe puede validar que el contenido del archivo está intacto utilizando el siguiente comando:

```
gpg --verify presupuesto-1002_firm.txt
```

### 3.6. Enviando las claves al repositorio de claves en Internet

Cuando nosotros creamos una clave o firmamos una clave pública ajena es importante permitirle a todos los demás usuarios de gpg estos nuevos cambios. De esta forma la red de confianza se agranda y uno puede intercambiar información (encriptandola y firmandola) con personas que no conocemos directamente.

Para hacer esto se puede utilizar el siguiente comando:

```
gpg --key-server <SERVIDOR> --send-keys [clave]
```

donde [clave] es un campo opcional (sino se especifica una clave, el gpg comenzará a enviar todas las claves que tiene en su base de datos de claves públicas), y la opción `--key-server <SERVIDOR>` es para indicarle la dirección del servidor en internet que actua como repositorio de claves <sup>3</sup>.

En el caso de que querramos descargar claves ajenas (o una en particular) se puede ejecutar el comando:

```
gpg --key-server <SERVIDOR> --recv-keys [clave]
```

las opciones actúan de la misma manera que al enviar claves.

Hay que tener en cuenta que los servidores que tienen los repositorios de claves públicas son *mirrors* unos de otros, por lo tanto, si uno envía una clave a un repositorio de claves, con los días (u horas) esta clave es transmitida a los demás repositorios.

### 3.7. Programas que soportan gpg

En la página de gpg (<http://www.gpg.org>) hay una hermosa tablita donde estan todos los programas que soportan correos firmados o encriptados con gpg, pero los que yo sé que lo soportan (porque los utilice o me contaron) son **evolution** (<http://www.ximian.com>), **KMail** (<http://www.kde.org>), **mutt** y **pine**.

Yo sí estuve mirando la hermosa tablita que te comento y entre los programas que soportan gpg hay muchos que son para Microsoft Windows (incluyendo Outlook) y otros sistemas operativos menos comunes como MacOS, OS/2, etc.

---

<sup>3</sup>Visita <http://www.gnupg.org> para saber las direcciones de los diferentes repositorios de claves

### *CAPÍTULO 3. CÓMO SE USA EL GPG*

---

Lo super interesante de utilizar estos programas es que uno no tiene que acordarse de todas estas opciones y argumentos del gpg ;-). De lo que sí uno no puede escaparse es de crear la clave, bah! por ahora ...

## Apéndice A

# Recursos utilizados

- Este documento esta basado en lo escrito por Mick Bauer en su columna de “Paranoid Penguin” de la revista Linux Journal (Jul/Oct 2001)
- La página del proyecto GnuPG
- Linux Documentation Project

# Índice general

<b>1. Introducción</b>	<b>1</b>
1.1. ¿Qué hace el gpg?	2
1.1.1. ¿Para qué te sirve a vos gpg?	2
1.2. ¿Cómo funciona la criptografía de claves públicas?	3
1.3. La Red de Confianza	4
<b>2. Instalación de gpg</b>	<b>5</b>
2.1. Desde los fuentes	5
2.2. Paquetes binarios	6
2.2.1. Con RPM	6
2.2.2. Con DEB	6
<b>3. Cómo se usa el gpg</b>	<b>7</b>
3.1. Creando la clave	7
3.1.1. Los archivos físicos	10
3.2. El certificado de revocación	10
3.3. Preparando las cosas para armar la red de confianza	11
3.4. Encriptado y Desencriptando información	13
3.5. Firmando y Validando información	13
3.6. Enviando las claves al repositorio de claves en Internet	14
3.7. Programas que soportan gpg	14
<b>A. Recursos utilizados</b>	<b>16</b>

# Índice alfabético

```
gpg --armor --export, 11
gpg --armor --export --output
    dcoletti.asc, 11
gpg --gen-key, 7
gpg --import ./dcoletti.asc, 12
gpg --key-server <SERVIDOR> --recv-keys
    [clave], 14
gpg --key-server <SERVIDOR> --send-keys
    [clave], 14
gpg --output /home/dcoletti/shadow
    --decrypt shadow.gpg, 13
gpg --output cert_revoc_arch.asc
    --gen-revoke <nombre_de_clave>,
    11
gpg --output presupuesto-1002_firm.txt
    --clearsign prespuesto-1002.txt,
    13
gpg --verify presupuesto-1002_firm.txt,
    13
```

## instrucciones

```
gpg --armor --export, 11
gpg --armor --export --output
    dcoletti.asc, 11
gpg --gen-key, 7
gpg --import ./dcoletti.asc,
    12
gpg --key-server <SERVIDOR>
    --recv-keys [clave], 14
gpg --key-server <SERVIDOR>
    --send-keys [clave], 14
gpg --output /home/dcoletti/shadow
    --decrypt shadow.gpg, 13
gpg --output cert_revoc_arch.asc
    --gen-revoke <nombre_de_clave>,
    11
```