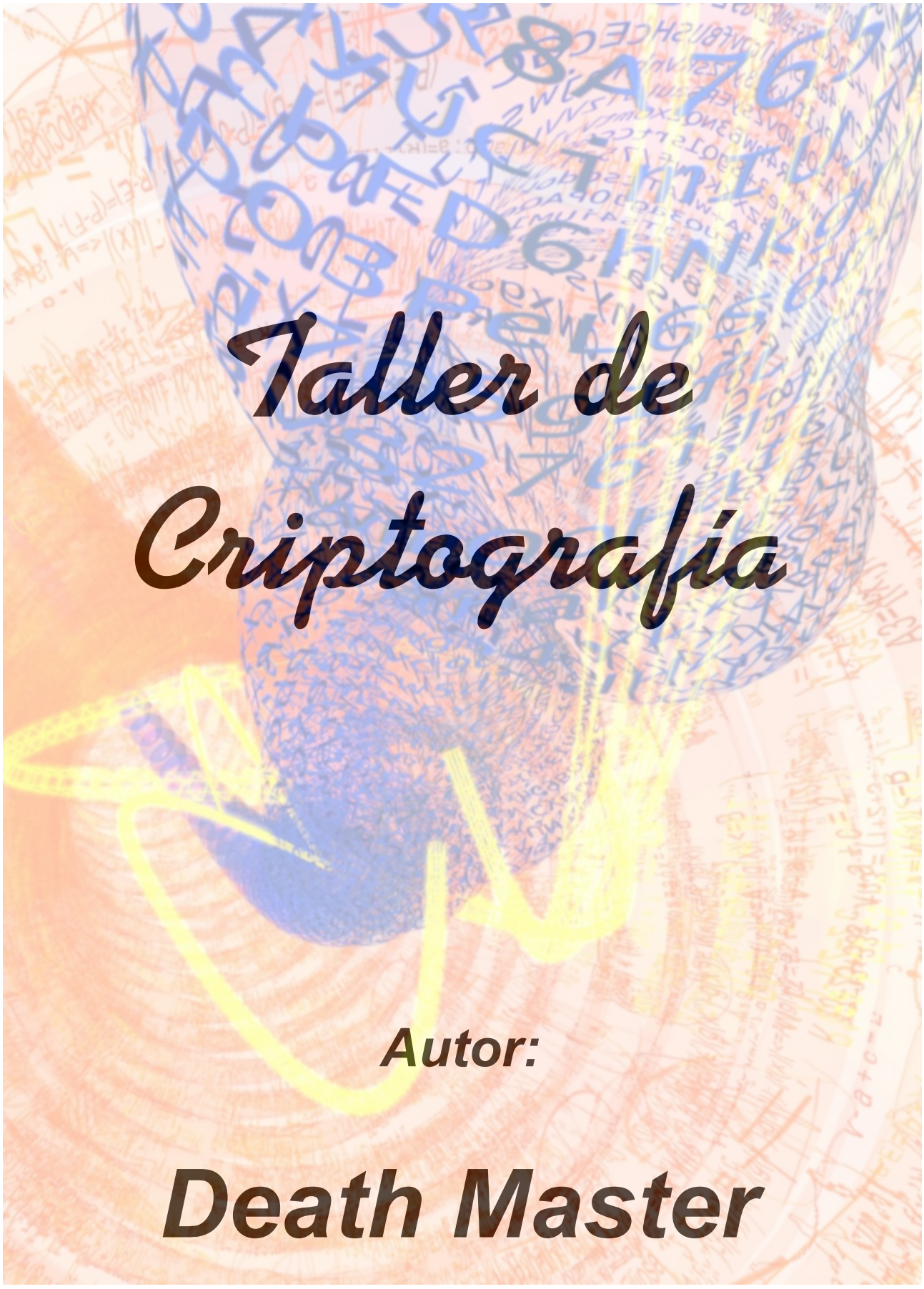

Taller de
Criptografía
Autor:
Death Master

Introducción

"El que confía sus secretos a otro hombre se hace esclavo de él."

Baltasar Gracián

Bienvenidos una vez más al Taller de Criptografía, que por fin es libre.

Este texto es una recopilación de los cinco capítulos que componen el "Taller de Criptografía", de los cuales los cuatro primeros fueron publicados en la revista "PC Paso a Paso: Los Cuadernos de Hack X Crack" en los números 27, 28, 29 y 30 respectivamente. El quinto capítulo tenía prevista su publicación en el número 31, pero debido a problemas económicos, la revista dejó de editarse y dicho capítulo nunca llegó a ver la luz... hasta ahora.

En el tiempo transcurrido desde su publicación, muchos de los datos (sobre todo en lo que a versiones de software se refiere) del presente documento han quedado desfasados. Aún así, he querido publicarlo exactamente igual a como fue editado en la revista, para que quede constancia exacta de lo que fue este Taller de Criptografía. Espero que sepáis comprender y perdonar esos pequeños fallos.

Quisiera dedicar este texto a todos los compañeros y compañeras que, gracias a los foros de Hack X Crack, pude conocer: **AnFe**, **CrashCool**, **DiSTuRB**, **el_chaman**, **FIFINA**, **Grullanetx**, **Kado**, **kurin**, **LorD_Darkness**, **moebius**, **Moleman**, **neofito**, **NeTTinG**, **oderty**, **okahei**, **Pintxo**, **Popolous**, **Storm666**, **TaU** (gracias por la imagen de la portada :-D), **TuXeD**, **Vic_Thor**, **Yorkshire**, **zurg**... y a todos los usuarios del foro. Muchas gracias a todos vosotros por todo lo que me habéis enseñado.

También quiero dedicárselo a una persona muy especial para mí: mi novia Laura. Gracias... por todo.

Death Master

Índice de contenidos

1.- El sistema PGP

(Publicado en "PC Paso a Paso: Los cuadernos de Hack X Crack" número 27, 14 de Febrero de 2005)

Bienvenidos al Taller de Criptografía. Si habéis seguido la revista de forma regular, recordaréis sin duda artículos como el de envenenamiento de caché ARP de moebius, el de sniffers del número anterior y muchos otros que, de una u otra forma, nos han recordado lo precaria que es la seguridad de nuestros datos, especialmente en el ámbito de redes locales: cualquiera con unos mínimos conocimientos técnicos y un poco de paciencia puede leer nuestros correos como si tal cosa, robar contraseñas... Por supuesto, todo esto puede evitarse con un diseño de red (tanto de política como de componentes físicos y su diseño lógico) adecuado... pero siempre depende de "otros" (a no ser que seamos el administrador de esa red, claro) y hoy en día accedemos a muchos datos sensibles a través de Internet y esas conexiones se realizan desde muchos sitios, no solamente desde nuestra casa. ¿Qué podemos hacer como simples usuarios desde nuestro terminal? Por supuesto, echar mano de la criptografía.

¿Qué es lo que hace de la criptografía un método de protección de nuestra información tan bueno? Aunque la respuesta a esta pregunta tan sencilla es muy compleja... pero podemos extraer una primera idea muy importante: la criptografía se sirve de la infraestructura lógica y física que los ordenadores nos brindan para proteger la información en sí. Otros métodos (IDS, Firewalls, diseños de red...) se basan en modificar de una u otra forma esa misma infraestructura, pero dejando la información inalterada. El principal problema de esta segunda opción es que resulta técnicamente más costosa, y que una vez encontremos una debilidad en estos sistemas de protección, la información estará igual de expuesta que si no existieran. Al cifrar nuestros datos, estamos usando un método de protección de la información en sí, que es matemáticamente muy complejo y poderoso, pero que de cara al usuario es transparente y sencillo de usar.

En este Taller de Criptografía iremos viendo poco a poco cómo manejar este poderoso recurso que es la criptografía, y aprenderemos a incorporar sus funcionalidades a nuestras tareas cotidianas con el ordenador. La forma en que aprenderemos todo esto será eminentemente práctica, donde no faltarán las explicaciones técnicas, por supuesto, pero intentaremos que éstas aparezcan cuando sea necesario para la comprensión de nuestras prácticas... y por supuesto, paso a paso. ;-)

Codificar o cifrar, esa es la cuestión

Un concepto muy importante que genera confusión y que conviene aclarar desde el principio es la diferencia entre codificación y encriptación o cifrado. El hecho de codificar una información supone transformar esa información a una representación equivalente, de forma que el significado de la información en su representación original y en su representación transformada sea el mismo. El proceso inverso de la codificación es la decodificación y puede realizarse de forma directa y transparente. Ambos procesos pueden ser descritos mediante algoritmos (que generalmente suelen ser públicos o conocidos) de forma que cualquier persona pueda transformar una información en sus distintas representaciones.

> *Entendemos por algoritmo una descripción precisa de una sucesión de instrucciones que permiten llevar a cabo un trabajo en un número finito de pasos.*

Cuando hablamos de cifrado, nos referimos igualmente a una transformación que tiene por fin generar una representación equivalente de la información, pero con una gran diferencia: en el proceso o algoritmo de cifrado interviene un elemento llamado clave (del que más adelante hablaremos) y que resulta imprescindible para poder realizar el proceso de cifrado, descifrado o ambos. De esta forma, una información cifrada no puede ser reconstruida si no se conoce, además del algoritmo, la clave (o claves) criptográficas que protegen la información.

Imaginemos que tenemos un texto escrito en castellano con todos sus caracteres en minúscula. Si nosotros decidimos, en un alarde de creatividad (:-P), escribir de nuevo ese texto pero con todos sus caracteres en letras mayúsculas, habríamos realizado una codificación del mismo en mayúsculas. Cualquier persona que conozca el alfabeto, aunque fuera incapaz de entender lo que dice, sería capaz de decodificarla de nuevo a minúsculas.

Ahora imaginemos que ese mismo texto lo traducimos a algún otro idioma, por ejemplo a francés. En este caso la transformación de la información la realizamos a través de un diccionario castellano-francés, que actuaría como clave, y obtenemos un texto cifrado en francés. Si alguien quisiera recuperar la información original en castellano, necesitaría un diccionario francés-castellano para descifrar la información.

Ya, ya sé que si hablas francés y castellano no necesitas diccionario, se trata únicamente de un ejemplo para que entendamos de forma intuitiva la diferencia entre codificar y cifrar.

> *Cuando hablamos de criptosistemas informáticos, nos referimos al texto original (el texto "legible") como texto en claro. El texto una vez cifrado se denomina criptograma.*
> *El proceso para convertir texto en claro a criptograma es lo que conocemos como cifrado o encriptación, y el proceso que convierte criptogramas en textos en claro lo llamamos descifrado o desencriptación.*

Pretty Good Privacy

Echemos la vista atrás, concretamente al año 1991 en Estados Unidos. El "boom" informático estaba en pleno apogeo, los nuevos y potentes (por aquel entonces, claro) 80486DX de 32 bits y con caché de nivel 1 incorporada llevaron a los ordenadores personales una potencia hasta entonces desconocida, y posibilitaron que cierto tipo de aplicaciones que hasta entonces no habían llegado al mercado doméstico lo hicieran. Uno de ellos la criptografía.

Cuando los rumores sobre posibles leyes para prohibir la criptografía comenzaban circular, un programador llamado Philip Zimmermann programó un software gratuito que mezclaba los mejores algoritmos de cada tipo (más adelante hablaremos sobre esto) y permitía a cualquiera con un ordenador personal hacer uso de una criptografía muy poderosa, equiparable a la de cualquier gobierno. Éste programa se llamó PGP, acrónimo de Pretty Good Privacy, y hoy en día sigue siendo el referente en su campo.

La historia de PGP es muy curiosa, y os animo a leer más sobre ella... aunque éste no es el sitio. Estoy seguro que google os puede echar una mano. ;-)

En esta primera parte del taller de criptografía vamos a valernos de PGP bajo Windows (aunque también puede ser seguido desde Macintosh) para aprender a usar y comprender la criptografía. Más adelante veremos otra implementación del estándar OpenPGP que es casi tan famosa, tiene versiones para casi cualquier sistema operativo y además es software libre: GnuPG.

> *OpenPGP es el estándar que surgió a partir de PGP, y que marca las pautas de compatibilidad que permiten a las distintas implementaciones ser completamente compatibles. Podéis consultar la descripción de este estándar en el RFC 2440 que podéis encontrar en:*
> *ftp://ftp.rfc-editor.org/in-notes/rfc2440.txt*
> *Lamentablemente no he encontrado ninguna traducción de este RFC al castellano.*

Consiguiendo e instalando PGP

Aunque históricamente desde casi sus inicios han existido dos versiones de PGP (la internacional, centralizada en el sitio http://www.pgpi.org/ y la estadounidense de la PGP Corporation http://www.pgp.com/), hoy en día prácticamente todo el mundo usa la versión PGP estándar. Aunque la PGP Corporation ha crecido muchísimo, siguen fieles a una de sus máximas desde sus inicios: ofrecer una versión gratuita de PGP para uso individual, educacional y en general cualquier actividad sin ánimo de lucro. En el momento de escribir este artículo la última versión es la 8.1, que podemos descargar de este enlace:

http://www.pgp.com/downloads/freeware/

Otra de las máximas de PGP (que no siempre han mantenido, por cierto, pero por suerte actualmente sí está vigente) es ofrecer el código fuente de PGP para que cualquier persona pueda examinarlo. Si sois los más paranoicos de vuestro barrio y tenéis buenos conocimientos de programación, podéis echar un vistazo al código en el siguiente enlace:

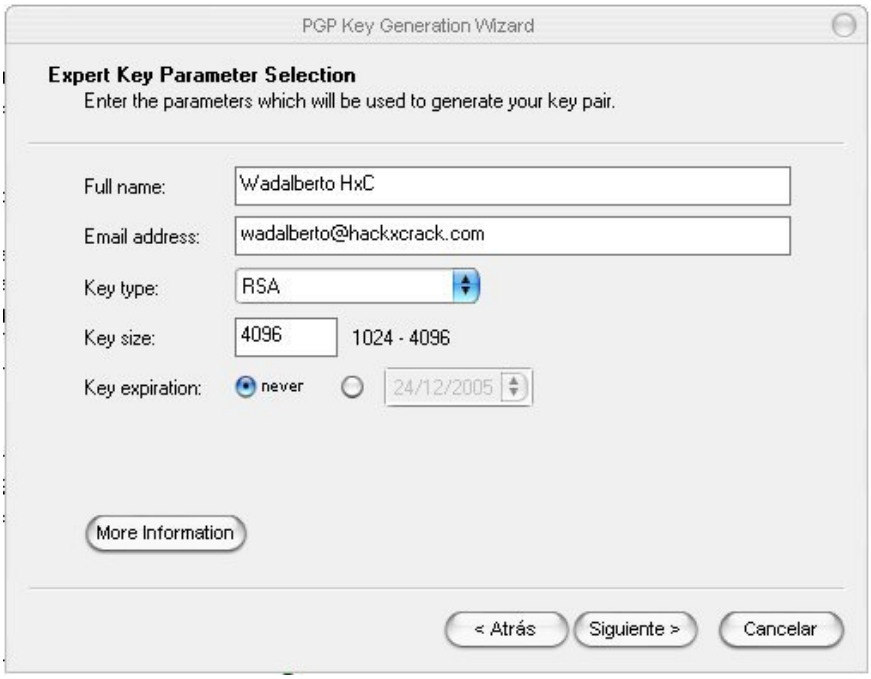
http://www.pgp.com/downloads/sourcecode/index.html

Si decides bajarte el código fuente de PGP, es MUY importante que leas la licencia a la que dicha descarga está sujeta.

Una vez bajado, procederemos a instalarlo. Las pantallas que veremos durante la instalación son las siguientes:

1) Welcome. Pantalla de bienvenida.
2) License Agreement. Debéis leer y aceptar (o no, en cuyo caso no podréis instalar el software) la EULA de PGP.
3) Read Me. Es conveniente que al menos echéis un vistazo al léeme del programa.
4) User Type. Esta es la primera pantalla interesante... seleccionaremos la opción “No, I'm a New User” y continuaremos.
5) Install Directory. Seleccionamos el directorio de instalación que queremos usar.
6) Select Components. En esta pantalla podremos configurar los distintos componentes adicionales de PGP que podemos instalar, tales como plugins para software de uso común, PGPdisk (solamente para versiones registradas) y demás. Si queréis instalar alguno, podéis hacerlo, pero no vamos a utilizarlos, al menos por el momento.
7) Start Copying Files. Veremos un resumen de lo que se va a instalar, y si estamos de acuerdo con todo, procedemos a su instalación.
8) Install Complete. Una vez finalizada la instalación es necesario reiniciar el ordenador para que el software se cargue correctamente.

Una vez reiniciado el ordenador, PGP iniciará automáticamente el asistente de claves (PGP Key Generation Wizard) mediante el cual crearemos nuestro par de claves único y personal. Pulsamos en el botón “Expert” (¿acaso no nos gusta exprimir todo al máximo? ;-P) y estaremos ante el menú pormenorizado de generación de claves.



Recordemos lo que sabemos sobre claves criptográficas: son el elemento más importante de un criptosistema, su alma. Gracias a las claves, un mismo algoritmo de cifrado puede ser usado por multitud de personas sin que el criptosistema coincida. La clave sería el “diccionario” que nos permitiría pasar de texto en claro a criptograma y viceversa.

En el campo “Full Name” introducimos nuestro nombre (o nick) y en “Email address” nuestra dirección de correo electrónico. Hasta aquí todo bien, pero llegamos al campo “Key Type” y nos encontramos tres opciones: Diffie-Hellman/DSS, RSA y RSA Legacy. Son los distintos tipos de algoritmos que PGP nos brinda para generación de claves asimétricas... es el momento de hablar de ellas.

Algoritmos de cifrado asimétricos

Imaginemos que no disponemos de un diccionario castellano-francés/francés-castellano y que únicamente tenemos uno castellano-francés y otro francés-castellano. Cuando nosotros traduzcamos nuestro texto de castellano a francés necesitaremos uno de los diccionarios, y cuando queramos realizar el proceso inverso necesitaremos el otro. Este sería el concepto de claves de cifrado asimétricas.

En un algoritmo criptográfico asimétrico existen dos claves complementarias, de forma que lo que una cifra puede ser descifrado por la otra y viceversa. A una de ellas le denominaremos clave pública o de cifrado, y a la otra clave privada o de descifrado. Dado que ambas claves deben ser complementarias, ambas se generan en el mismo momento y ambas son necesarias para que el criptosistema funcione. Más adelante veremos el funcionamiento concreto de las claves asimétricas.

Existen muchos algoritmos asimétricos basados en diversos problemas matemáticos, pero PGP nos ofrece dos posibilidades:

DH/DSS (Diffie-Hellman/Digital Standard Signature). Este algoritmo se basa en el problema matemático del "logaritmo discreto del grupo multiplicativo de un campo finito" (obviamente no necesitáis saber qué demonios significa esto :-D) y fue ideado por Dress. W. Diffie y M.E. Hellman en 1976 (podéis consultar el documento [DIH76] para más referencias), aunque no tuvo implementación computacional hasta 1985, cuando ElGamal logró demostrar que ésta era posible en su documento [ELG85]. En 1991 el NIST propuso DSS como ampliación a DH para convertirlo en un criptosistema completo (DH carecía de un sistema de firma... más adelante hablaremos de las firmas criptográficas). Así nació DH/DSS.

RSA (Rivest-Shamir-Adleman). Este algoritmo fue ideado en 1978 por Ron Rivest, Adi Shamir y Leonard Adleman y descrito en el documento [RSA78]. Basa su potencia en el problema matemático de la factorización entera (no, tampoco es necesario entender eso). El algoritmo fue patentado, pero la patente expiró en el año 2000, por lo que hoy en día puede ser usado libremente.

> *Aunque suene casi a ciencia ficción, existe un algoritmo para computadores cuánticos que ha sido probado y demostrado por IBM y que permitiría reducir enormemente la complejidad del problema de la factorización entera. Éste algoritmo es conocido como "Algoritmo de Shor" y aunque facilitaría enormemente la ruptura de claves RSA, seguiría siendo un problema muy complejo para claves grandes... y requeriría un ordenador cuántico bastante potente. Podéis saber más sobre el algoritmo de Shor en:*
> *http://en.wikipedia.org/wiki/Shor's_algorithm*

Si os estáis preguntando porqué he dicho que PGP ofrece dos posibilidades cuando en realidad en nuestro menú de generación de claves podemos elegir tres... pues entonces es que estáis bastante atentos ;-). En realidad solamente hay dos posibilidades: DH/DSS y RSA. La tercera opción (RSA Legacy) es una versión "vieja" de RSA que solamente se usa para mantener compatibilidad con claves de versiones antiguas de PGP, pero que a nosotros no va a interesarnos, puesto que no implementa claves de revocación, múltiples subclaves y otros aspectos que el RSA actual sí soporta.

Así pues, tenemos que decidir qué algoritmo elegimos. Eso es una decisión que dejo en vuestras manos, aunque sí diré que tradicionalmente (y debido a la patente que hasta el año 2000 atenazaba al algoritmo RSA) se ha usado DH/DSS, y que de hecho es el que más gente usa... y también diré que yo uso RSA. Pero cada cual que elija el que más rabia le dé, es indiferente para el seguimiento de este taller.

En "Key size" debemos indicar el tamaño de la clave que deseamos generar. Cuando más grande sea nuestra clave, muchísimo más compleja resultaría de romper en un momento dado. Tanto RSA como DH soportan claves de hasta 4096 bits, aunque cabe destacar que DSS, el algoritmo de firma de DH, solamente soporta hasta 1024, por lo que solamente podremos crear claves DH/DSS de como mucho 4096/1024 bits. Mi recomendación de friki-paranoico es que elijáis los 4096 bits.

Como curiosidad: la relación que guarda el tamaño de la clave con la complejidad de ésta es exponencial, pues dada una clave cualquiera, al aumentar un bit el tamaño de la misma, duplicaríamos su complejidad. Así, una clave de 1025 bits es el doble de compleja que una de 1024. Podéis echar las cuentas de cómo de compleja es una clave de 4096 bits... :-P

Ahora nos encontramos con el campo "Key expiration", donde podremos indicar, si la deseamos, la fecha de caducidad de nuestra clave. Ésto puede resultar útil cuando queremos definir claves por un período de tiempo, por ejemplo. En nuestro caso no nos interesará, aunque si alguien quiere utilizar esta opción, no alterará el desarrollo de las prácticas. Por fin, pulsamos siguiente.

El passphrase

Nos encontramos en la pantalla de creación del passphrase (que es a password lo que frase a palabra). Debemos definir la contraseña que protegerá las acciones que nuestra clave privada es capaz de realizar. Como bien sabemos, una cadena siempre se rompe por su eslabón más débil, por lo que una clave de 4096 bits que caiga en manos inapropiadas y con un passphrase predecible deja todo el criptosistema al descubierto.

Mi recomendación es que uséis una contraseña SEGURA. Al menos de 8 ó 10 caracteres de largo, con letras mayúsculas, minúsculas, números y si es posible símbolos. La mía es de 25 caracteres, y cuando te acostumbras no es tan incómodo. :-P

Al pulsar siguiente empieza el proceso de generación de la clave propiamente dicho, tras el cual la clave es completamente utilizable.

El proceso de generación de la clave tiene un componente aleatorio que viene dado por la entropía que introduce el usuario en el sistema: movimientos del ratón, uso de los periféricos de entrada/salida... esto es así porque los ordenadores NO pueden generar números auténticamente aleatorios (únicamente pseudoaleatorios), lo que haría que una clave generada únicamente por el ordenador pudiera ser predecible.

Las opciones de PGP

Antes de meternos en faena, vamos a echar un vistazo a las opciones de PGP pulsando con el botón derecho en el icono PGPtray y seleccionando "Options".

En la pestaña "General" vemos en primer lugar las opciones estándar, donde si queremos podremos añadir una línea de comentario a los mensajes o ficheros generados con PGP. En el apartado "Single Sign-On" tenemos la opción de guardar en caché los passphrases del usuario, con el fin de evitar tener que estar tecleándolos una y otra vez. Por defecto esta opción está activada. Pues bien, recomiendo ENCARECIDAMENTE desactivar esta opción, por evidentes motivos de seguridad. El apartado de "File Wiping" lo dejamos descansar... de momento, porque es MUY interesante y más adelante hablaremos de él.

En la pestaña "Files" podemos elegir la localización que queremos para nuestro anillo de claves.

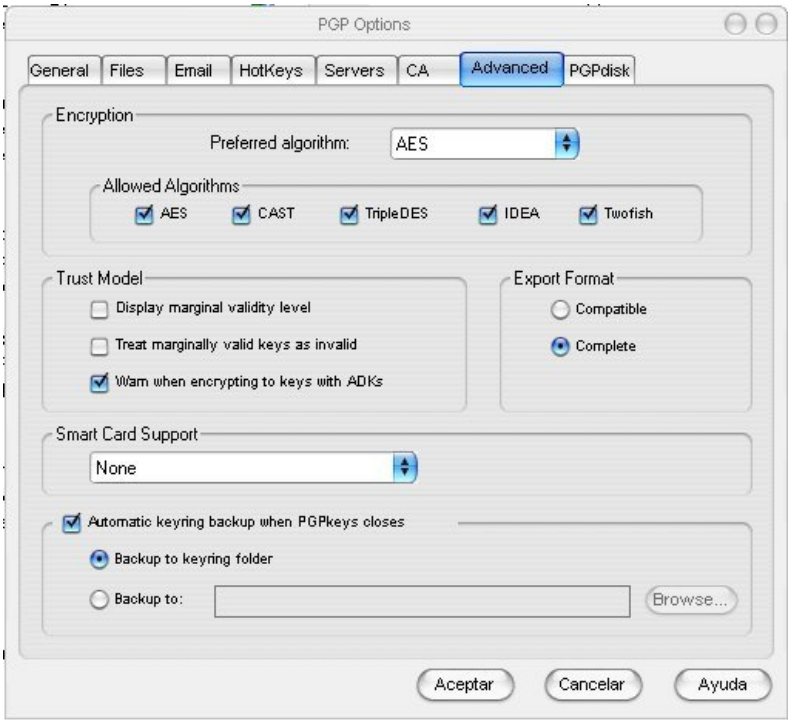
En los sistemas OpenPGP las claves, subclaves y firmas se organizan en un sistema llamado anillo de claves y que permite la interacción de unas con otras (identificación de firmas, revocación de elementos...) así como el uso simultáneo de las mismas (por ejemplo, cifrar un archivo a varios destinatarios).

La pestaña "Email" nos permite configurar diversos aspectos del comportamiento de PGP con los MUAs (Mail User Agents), pero de momento este menú no lo tocaremos, pues el tema de PGP y correo electrónico corresponde a otro artículo. En la pestaña "HotKeys" podemos definir atajos de teclado para diversas operaciones comunes de PGP... si sois un poco vagos quizá os guste definir unos cuantos atajos, pero yo personalmente no uso ninguno.

En la pestaña “Servers” encontramos los diversos servidores de claves PGP que el programa usará para sincronizar las claves, firmas y revocaciones con las de nuestro anillo. De esta forma, cuando recibamos un correo de una clave que no está en nuestro anillo, podremos conectarnos “en caliente” al servidor y obtener esa clave para verificar la autenticidad del mensaje (siempre y cuando el usuario de esa clave haya subido la misma al servidor, claro... pero de eso hablaremos más tarde). La pestaña “CA” corresponde a la configuración de las Autoridades de Certificación... eso no lo tocaremos, pues se escapa bastante del temario de este artículo.

Llegamos por fin a la pestaña “Advanced”, donde más “chicha” vamos a encontrar... :-)

En primer lugar nos encontramos con el apartado “Encryption”, donde configuraremos las opciones del cifrado simétrico de PGP.



> *¿Qué? ¿Cómo? ¿Pero no habíamos quedado que PGP usaba claves de cifrado asimétricas? ¡Tú me estás engañando! ¡Po dió!*
> *Tranquilos... tiene su explicación. En realidad PGP se sirve no de uno ni dos, sino de TRES tipos de cifrado para poder realizar sus distintas funciones.*
> *Más tarde explicaremos el mecanismo concreto que explica este aparente contrasentido, aunque como anticipo diré que en realidad todo lo que cifremos con PGP se hará con claves simétricas.*

¿Y qué es el cifrado simétrico? Imaginemos el ejemplo de los diccionarios... e imaginad que ahora sí existe un diccionario castellano-francés/francés-castellano. Sería la única clave necesaria para cifrar y descifrar el texto. Así pues, en un criptosistema simétrico, la clave es única y cumple las funciones de cifrado y descifrado.

> *Los criptosistemas simétricos se dividen a su vez en dos tipos: los de cifrado de bloque, que cifran el texto en claro en unos bloques con tamaño prefijado (por ejemplo 32 bits ó 64 bits); y los de cifrado de flujo, que cifran de forma continuada bit a bit o byte a byte.*

Las opciones que encontramos son las siguientes:

AES (Advanced Encryption Standard), 2000. También conocido como algoritmo Rijndael, es el ganador del concurso que en 1977 convocó el NIST para definir el nuevo estándar de cifrado simétrico (por ello, al ganar pasó a llamarse AES). Fue ideado por los belgas Vincent Rijmen y Joan Daemen y puede actuar como algoritmo de cifrado de bloques o de flujo. Soporta claves de hasta 256 bits.

CAST (Carlisle Adams - Stafford Tavares), 1997. Existen dos versiones de este algoritmo: la 128 y la 256 (indica el tamaño máximo de clave que maneja), y en ambos casos se trata de un algoritmo de cifrado en bloques de 128 bits. El funcionamiento interno de este algoritmo es uno de los más complejos (y por tanto, computacionalmente más lentos) de entre los que PGP nos brinda.

TripleDES (Triple Data Encryption Standard), 1995. En realidad TripleDES no es un algoritmo en sí mismo, sino que resulta de la aplicación tres veces con distintas claves del veterano algoritmo DES de 1976 (el antecesor de AES). Maneja longitudes de clave de hasta 168 bits efectivos (más 24 de paridad), y aunque no es muy complejo, sí es muy rápido en su cálculo.

IDEA (International Data Encryption Algorithm), 1990. Se trata de un algoritmo de cifrado en bloques de 64 bits ideado por Xuejia Lai y L. Massey que soporta claves de longitud de hasta 128 bits. Es importante mencionar que este algoritmo está patentado y la patente aún es vigente en Estados Unidos así como en Europa (motivo por el cual algunas implementaciones OpenPGP, como por ejemplo GnuPG, no lo implementan).

Twofish, 1998. Este algoritmo es la evolución de uno llamado Blowfish, y fue uno de los finalistas del concurso AES del NIST. Twofish es un algoritmo de cifrado en bloques de 64 bits que maneja claves de hasta 256 bits. Este algoritmo destaca por ser el más rápido en su ejecución con mucha diferencia, motivo por el cual es bastante usado (por ejemplo en SSH).

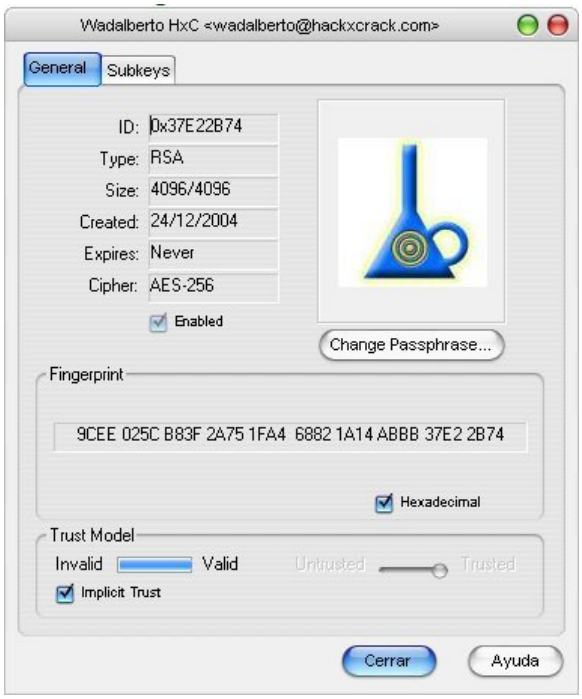
Conviene tener todos activados (en “Allowed algorithms”), pero hay que elegir uno preferido... bien, aquí yo recomiendo usar AES, por compatibilidad y por potencia, si bien Twofish me gusta también bastante. Las demás opciones de esta pantalla podemos dejarlas como vienen perfectamente.

La pestaña “PGPdisk” solamente podrá ser configurada si tienes una versión registrada de PGP, por lo que aquí la obviaremos.

La clave PGP

Ahora que ya tenemos nuestra clave creada y que hemos podido personalizar las opciones de PGP... es el momento de profundizar en la clave.

Lo primero que tenemos que hacer es acceder al administrador de claves de PGP, lo cual haremos pulsando con el botón derecho en el icono PGPtray y seleccionando la opción PGPkeys. Al seleccionar y pulsar con el botón derecho del ratón sobre nuestra clave, veremos el menú contextual de la misma. Sí, sé que son muchas cosas... pero de momento solamente pulsaremos en “Key Properties” para poder entrar en detalle.



La fotografía en PGP es opcional y además no suele usarse demasiado. Se puede añadir mediante el menú “Keys, Add, Photo”. La que he incluido en la clave de ejemplo es un pequeño guiño a mis amigos del foro hackxcrack... y ya de paso, os invito a que os paséis por él si no lo habéis hecho ya. Os aseguro que aprenderéis mucho. http://www.hackxcrack.com/phpBB2/

Analicemos los campos que vemos en este menú porque son muy importantes:

ID: En el campo ID veremos un número hexadecimal (0x...) que identifica a nuestra clave y resulta muy útil para realizar consultas al servidor de claves. Aunque es extremadamente complicado (una posibilidad entre más de cuatro mil millones) es posible que este ID coincida con el de otra clave, por lo que este campo NO sirve para identificar a la clave de forma unívoca, y simplemente lo usaremos como referencia.

Type: El tipo de clave asimétrica que elegimos en el momento de la generación.

Size: El tamaño de nuestra clave en formato xxxx/yyyy donde xxxx representa la longitud de cifrado e yyyy representa la longitud de la firma.

Created: Fecha de creación de la clave en cuestión.

Expires: Fecha de expiración de la clave. Por defecto, y si no indicasteis explícitamente lo contrario, será "Never".

Cipher: Algoritmo preferido de cifrado simétrico. Generalmente será AES-256.

Casilla "Enabled": Nos permite, cuando es una clave externa (llamamos clave externa a aquella que únicamente tiene componente de clave pública, careciendo de la privada), desactivar la misma, bien porque haya caducado, bien porque ya no sea segura, bien porque no deseemos cifrar por equivocación a esa clave...

> *Hay que tener en cuenta que una clave con parte pública y privada no puede ser desactivada en ningún caso.*

Botón Change Passphrase: Al pulsar sobre este botón, y previa autentificación mediante el actual passphrase, es posible cambiar el mismo de la clave.

Fingerprint: Este concepto es uno de los más importantes en PGP. Como ya dijimos, aunque es complicado, es posible que el KeyID de dos claves distintas coincida, por lo que necesitamos un mecanismo que permita identificar de forma unívoca a cada una de las claves. Este mecanismo es el fingerprint, también denominado huella digital. Antes de continuar, si veis en este campo una serie de palabras, pulsad sobre la casilla "Hexadecimal" para verla como $DEITY manda. ;-)

> *Si con el KeyID las posibilidades de coincidencia eran de una entre cuatro mil millones, con el fingerprint, y dependiendo del algoritmo usado, son de una entre cifras del orden del sextillón (1x10^38) o del octillón (1x10^48).*

¿Y qué es el fingerprint? Simplemente es el hash de la clave pública... lo cual nos lleva a un poquito más de teoría, en este caso sobre las funciones hash.

Para entender qué es una función hash, vamos a hacer un poco de memoria... concretamente hasta nuestros tiempos del instituto. Imaginemos una función matemática de las de toda la vida, por ejemplo y=x^2.

La y sería la f(x), es decir, el valor dado en función de x. Ahora imaginemos que esa función es una máquina que transforma un valor de entrada que nosotros le damos en un valor de salida. Por ejemplo, si nosotros metemos en la función el valor 2, ésta nos devuelve su cuadrado: 4. Así ocurrirá para cualquier valor de x que nosotros le demos.

Ahora imaginemos que construimos una función que llamaremos la inversa de la anterior y que será y=raíz(x). Esta función f'(x) también podemos considerarla una máquina de transformar valores, y se da la propiedad de que si introducimos en nuestra segunda función f'(x) el valor de salida de nuestra primera función f(x) obtendremos el valor que introducimos en la primera. Por ejemplo, f(2)=4 y f'(4)=2.

¿Hasta ahora bien? Bueno, pues ahora vamos a echarle todavía un poquito más de imaginación. Imaginemos que construyo una función más compleja... en la que por el proceso pierda información. Por ejemplo, me construyo una función en la que elimino el número menos significativo de la entrada y lo que me quede lo elevo al cuadrado. Consideremos esta nueva función -que llamaré g(x)- de nuevo una máquina de transformar números. Introducimos el valor 23 y la función realiza su algoritmo: elimina el 2 y eleva el 3 al cuadrado. La salida de esta función sería 9. La cuestión es... ¿sería posible realizar una función g'(x) que fuera la inversa de la anterior? NO, porque en el proceso de la función se pierde una información que no puede ser recuperada a posteriori. Esto sería una función irreversible.
Un último esfuerzo imaginativo... ahora imaginemos que construyo una función irreversible lo suficientemente compleja como para que devuelva una salida bastante grande, lo suficiente como para que no coincida con facilidad dadas dos entradas distintas. Esto permitiría "resumir" distintas entradas a los valores que produce la función a partir de ellas... y esto sería una función hash.

Así pues, toda función hash tiene unas propiedades matemáticas que vamos a explicar de forma sencilla:

Unidireccional: Una función hash es irreversible, y dada una salida de la misma, es computacionalmente imposible recomponer la entrada que la generó.

Compresión: Dado un tamaño cualquiera de entrada para la función hash, la salida de la misma será de una longitud fija.

Difusión: La salida de la función hash es un resumen complejo de la totalidad de la entrada, es decir, se usan TODOS los bits de la misma.

Resistencia a las colisiones simples: Esta propiedad nos indica que dado un valor de entrada de la función hash, es computacionalmente imposible encontrar otra entrada que genere el mismo valor de salida.

Resistencia a las colisiones fuertes: Esta propiedad nos indica que es computacionalmente muy difícil encontrar dos valores de entrada que generen un mismo valor de salida.

En PGP se usan dos tipos de algoritmos hash:

MD5 (Message Digest 5), 1992. Este algoritmo, evolución del MD4 (que a su vez lo es del MD2), es uno de los más extendidos en Internet. Fue ideado por el matemático Ron Rivest (uno de los padres de RSA) y genera cadenas hash de 128 bits a partir de una entrada de cualquier tamaño.

> *Si eres usuario de GNU/Linux y alguna vez has descargado una imagen ISO de Internet, habrás visto que junto a los ficheros ISO suelen haber otros ficheros md5 (y a veces, un tercer fichero md5.asc). Ese fichero md5 contiene la cadena MD5 de la imagen ISO y permite comprobar que se ha descargado correctamente o que no ha sido alterada. El archivo md5.asc es una firma PGP del fichero MD5, de forma que nos aseguramos además que el fichero MD5 ha sido generado por el firmante.*
>
> *Existen muchas implementaciones de MD5 para que podemos usarla de forma cómoda en nuestro sistema. Los usuarios de Linux tienen el paquete md5sum, que va incluido con prácticamente cualquier distribución, mientras que los usuarios de Windows deben descargar alguna de las implementaciones de Internet, por ejemplo yo recomiendo ésta:*
> *http://www.fourmilab.ch/md5/*

SHA-1 (Secure Hash Algorithm), 1994. SHA-1 nació de manos del NIST como ampliación al SHA tradicional. Este algoritmo, pese a ser bastante más lento que MD5, es mucho más seguro, pues genera cadenas de salida de 160 bits (a partir de entradas de hasta 2^64 bits) que son mucho más resistentes a colisiones simples y fuertes. Hoy en día, es el estándar en PGP.

De esta forma, si deseamos añadir la clave de otra persona a nuestro anillo solamente debe proporcionarnos el fingerprint de su clave, pues nosotros nos encargaremos de buscar la clave en los servidores públicos, descargarla y verificar que el fingerprint es el mismo.

Trust Model: En este campo tenemos dos medidores: el primero de ellos mide la validez de la clave (por ejemplo, una clave caducada no sería válida), y el segundo mide el modelo de confianza que depositamos en esa clave (ésto puede resultar muy útil cuando tenemos gran cantidad de claves en nuestro anillo y al recibir un correo deseamos saber de un vistazo cuánto confiamos en esa clave). La opción "Implicit Trust" solamente es válida para claves con parte pública y privada, es decir, claves que nos pertenecen completamente aunque no hayan sido generadas en nuestra máquina. Es importante destacar que para poder otorgar confianza a una clave debemos haberla firmado previamente, pero de las firmas hablaremos más tarde.
Hay una pestaña más dentro del menú de propiedades de la clave, la pestaña "Subkeys". En ella podemos consultar las subclaves existentes dentro de la clave principal. Adicionalmente, y en el caso de claves con parte privada, podremos además añadir nuevas subclaves y recovar o eliminar las ya existentes.

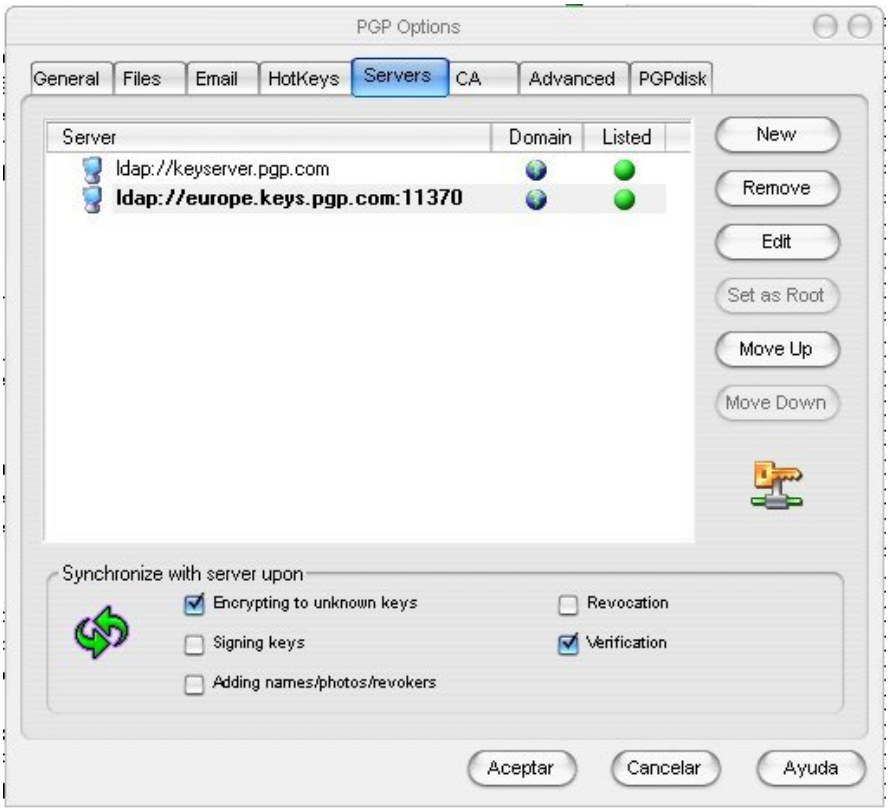
> *En realidad cuando generamos la clave tras la instalación de PGP, generamos también una subclave principal de cifrado, que es la que veremos en esta ventana. En PGP esta subclave principal (imprescindible para el correcto funcionamiento de la clave) se genera de forma automática, pero cuando veamos GnuPG en el siguiente artículo veremos que debemos generarla nosotros mismos a mano... lo cual puede ser un poco más engorroso pero también nos permite afinar hasta el límite el tipo de clave que queremos usar.*

Si deseamos usar más de un nombre y/o correo electrónico en nuestra clave, podemos añadirlos fácilmente. Solamente hay que pulsar con el botón derecho sobre la clave y seleccionar "Add" seguido de "Name". PGP nos pedirá el nuevo nombre y correo que deseamos añadir y se generará automáticamente. Para seleccionar cuál queremos que sea el nombre principal de la clave, debemos seleccionarlo con el botón derecho y elegir la opción "Set as Primary Name".

Exportando e importando: servidores de claves y copias de seguridad

Vamos a practicar un poco todo lo visto hasta el momento...

Lo primero que vamos a hacer es ir al menú de opciones de PGP (ya sabéis cómo) y al apartado "Servers". Una vez ahí, tendréis dos servidores: ldap://keyserver.pgp.com y ldap://europe.keys.pgp.com:11370.

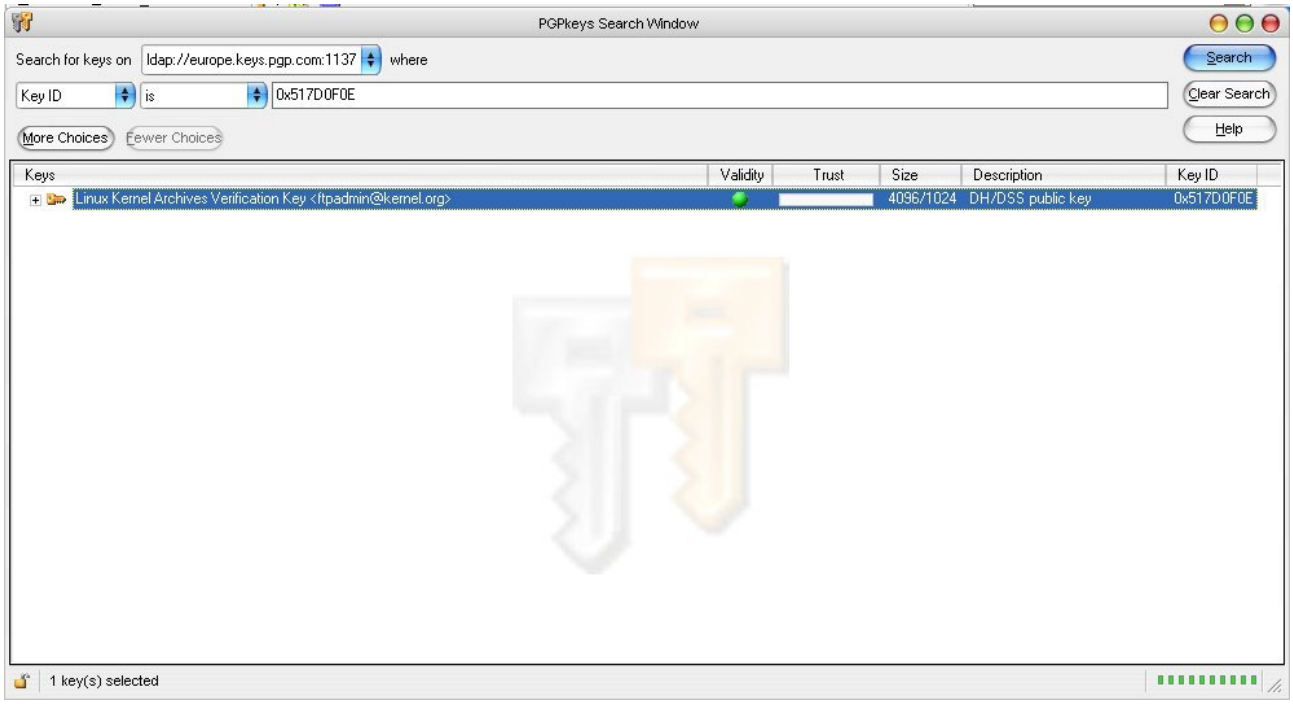


Seleccionad el segundo de ellos y pulsad en el botón “Set As Root” que hay en el menú de la derecha. Así cambiaremos nuestro servidor de claves principal.

> *El motivo para cambiar nuestro servidor principal a este segundo es que actualmente la PGP Corporation está realizando importantes cambios en el sistema de su servidor principal (el primero) y muchas de las claves no están disponibles al no haber sido verificadas aún por sus respectivos dueños. Para no tener problemas con claves que no están, usaremos el servidor europeo de claves.*

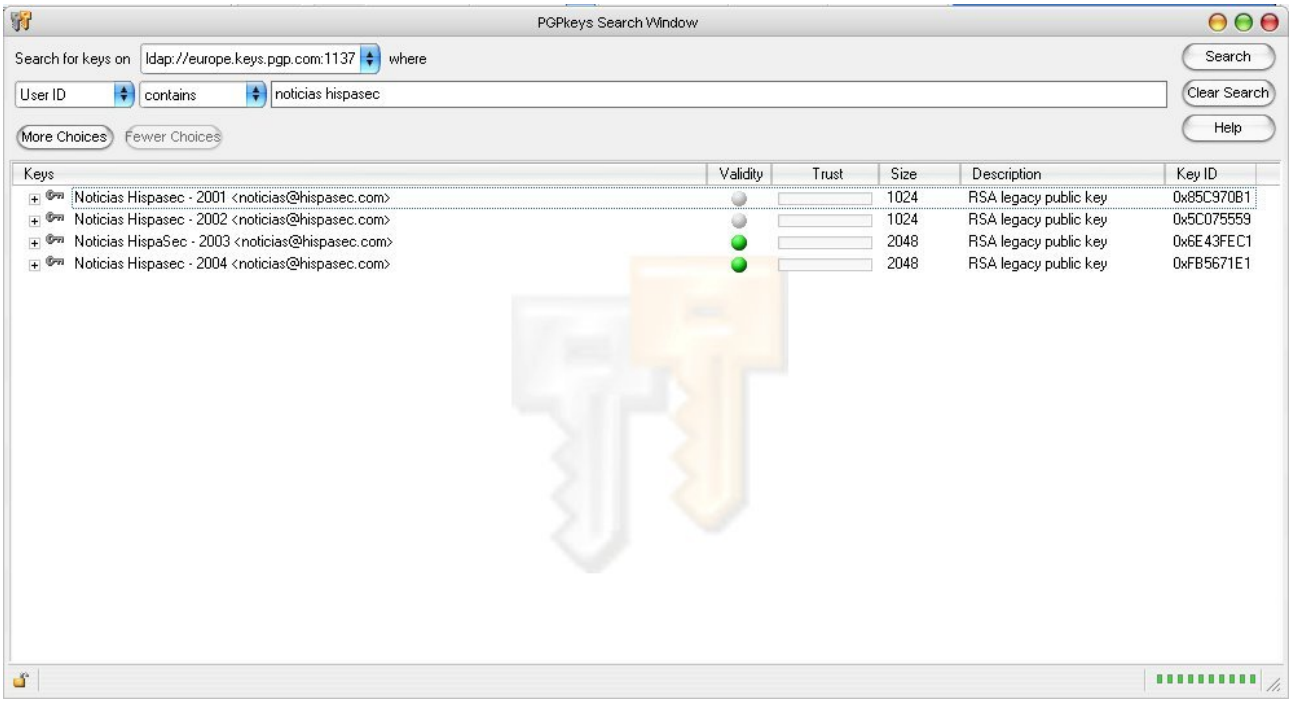
Ahora abriremos PGPkeys y pulsaremos en el menú “Server” seguido de “Search” (también es posible acceder directamente mediante el atajo ctrl+F). Éste es el menú de búsqueda de claves de PGP. Lo primero que hay que hacer es cambiar el servidor donde vamos a buscar las claves al europeo, y ahora vamos a realizar unas cuantas búsquedas con distintos criterios para familiarizarnos con el manejo de PGP.

En los criterios de búsqueda seleccionaremos “Key ID” e “is”. Introducimos “0x517D0F0E” y pulsamos search, tras lo cual aparecerá un único resultado en forma de clave pública a nombre de “Linux Kernel Archives Verification Key <ftpadmin@kernel.org>”. Pulsad con el botón derecho sobre la clave y seleccionad la opción “Import to Local Keyring”, momento a partir del cual esa clave pasará a formar parte de nuestro anillo de claves. Bien, ahora vamos a comprobar que vosotros y yo tenemos la misma clave, para lo cual nos valdremos del fingerprint de la misma:



C75D C40A 11D7 AF88 9981 ED5B C86B A06A 517D 0F0E

Si el fingerprint que podéis ver en la clave que hay en vuestro anillo es este mismo (que lo será), significa que la clave es la misma para ti y para mí.



Ahora vamos a realizar una segunda búsqueda cambiando los criterios. Ahora usaremos “User ID” y “contains”. Introduciremos “noticias hispasec” (aprovecho para mandar un saludo a toda la gente de hispasec por su magnífico trabajo) y realizaremos la búsqueda. El resultado en este caso serán cuatro claves RSA legacy. Las importaremos todas a nuestro anillo de claves.

> *En el momento de escribir este artículo (aún en el 2004) la búsqueda produce solamente cuatro resultados, con las claves de los años 2001, 2002, 2003 y 2004. Cuando realicéis estas prácticas aparecerá una clave más, la del 2005.*

Ahora podremos ver que nuestro anillo de claves está bastante menos solitario de lo que estaba antes. Ahora, borrad todas las claves de noticias hispasec que no estén ya vigentes. Para ello, seleccionad todas las claves manteniendo pulsada la tecla control y luego pulsad con el botón derecho y elegid la opción “Delete”. Las teclas también pueden ser copiadas, cortadas o pegadas como cualquier otro tipo de archivo.

Bien, tenemos ya nuestra clave creada y sabemos cómo podemos obtener nuevas claves de los servidores, pero ¿cómo pueden otras personas obtener nuestra clave? Pues para eso, obviamente, nuestra clave debe estar en el servidor de claves. Para publicar una clave en el servidor de claves, solamente hay que pulsar en el botón derecho sobre la clave y seleccionar “Sent To” y seleccionar a qué servidor enviar la misma. Si seleccionáis “Domain Server”, se enviará al servidor que tengáis configurado por defecto.

Probad a publicar en el servidor europeo de claves vuestra nueva clave recién creada y después probad a buscarla para poder estar seguros de que se ha publicado correctamente. Sería bueno que la buscarais por diversos criterios de búsqueda, para practicar un poco más con el motor de búsqueda de PGP.

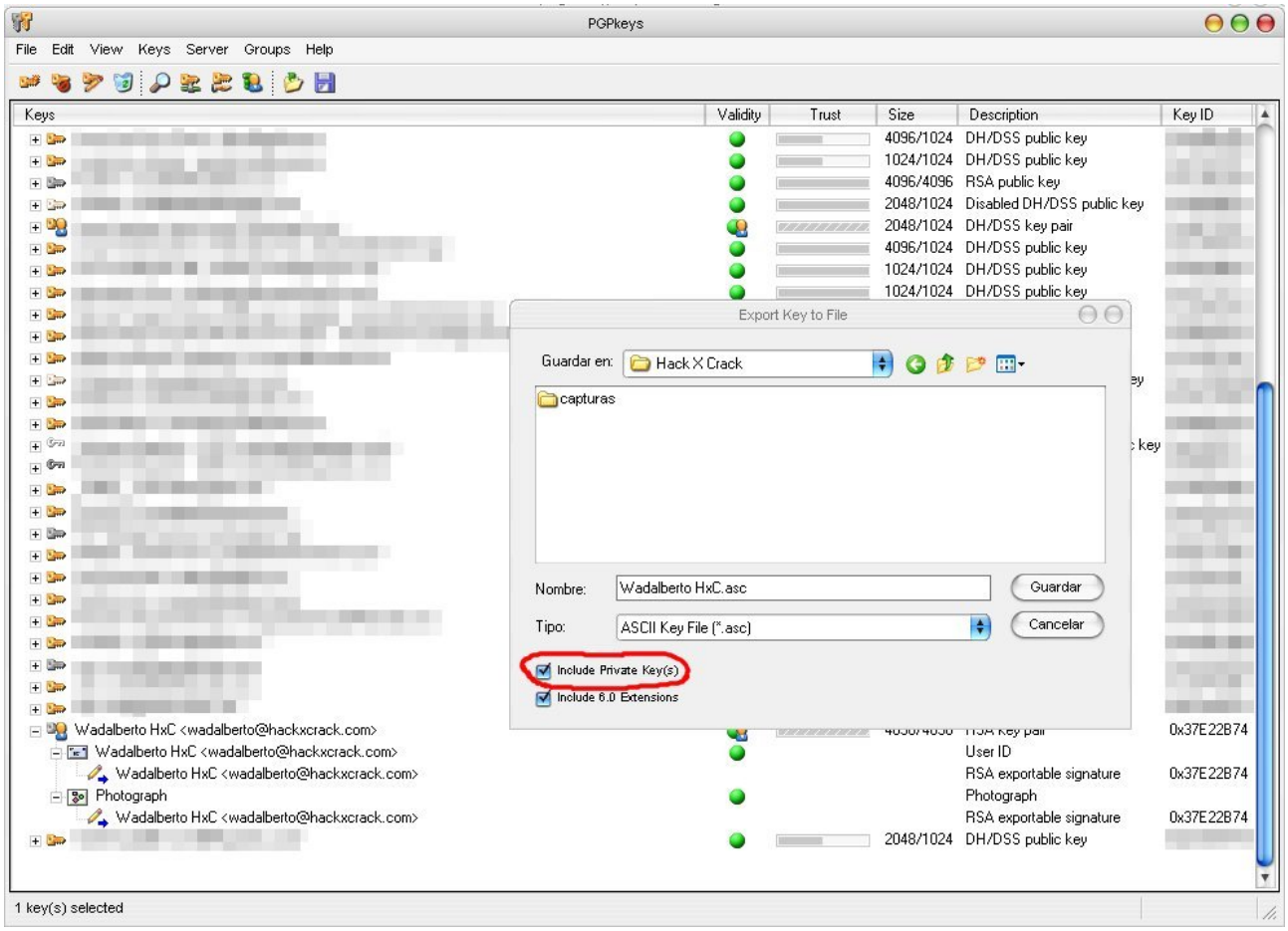
Otra opción importante es la de actualizar claves que ya tenemos en nuestro anillo de claves. Se puede hacer de dos formas: o bien buscando la clave mediante el motor de búsqueda de PGP, o bien usando la opción de actualización de PGP.

Para hacer uso de la opción de actualización de claves de PGP pulsad sobre la clave con el botón derecho y seleccionar “Update”. Aparecerá una ventana en la que podremos ver la clave y desarrollar el árbol de la misma para poder observar el estado en el que se encuentra en el servidor. Si decidimos que queremos incorporar los cambios existentes en la clave a nuestro anillo de claves local, debemos pulsar en “Import” y el software automáticamente gestionará la importación y actualización.

¿Cómo puede cambiar una clave una vez subida a un servidor? Pueden añadirse, eliminarse o revocarse subclaves, fotografías, firmas...

IMPORTANTE: Absolutamente todas las claves que se exportan a un servidor de claves y, por tanto, toda clave publicada en un servidor público, solamente consta de parte pública. Es imposible exportar un bloque privado a uno de estos servidores. ¿Porqué os cuento esto? Porque es MUY importante que mantengáis copias de seguridad de vuestras claves privadas de forma local (y a ser posible, no en el mismo disco duro) porque si perdéis vuestra clave privada, NO podréis recuperarla y NO podréis volver a descifrar nada que se cifre a esa clave ni generar más firmas (cof, cof, querido amigo mío AcidBorg).

¿Y cómo creamos copias de seguridad locales? Mediante la opción de exportación de claves. Al pulsar sobre la clave con el botón derecho y seleccionar la opción “Export” aparecerá una ventana donde debemos seleccionar el destino de la clave exportada. Es muy importante fijarse en la casilla “Include Private Key(s)” (marcada en la imagen), pues ésta es la opción que nos permite, al exportar una clave, incluir su parte privada y por tanto realizar una copia de seguridad de nuestra clave. Ésta opción está disponible únicamente para claves con parte pública y privada (obviamente) y está desmarcada por defecto.



El fichero generado será un archivo con extensión .asc que contendrá, en forma de armadura ASCII, la clave exportada (bien sea únicamente la parte pública o ambas partes). Uhm... un concepto nuevo a introducir: armadura PGP.

Una armadura... de caracteres

PGP puede generar principalmente tres tipos de ficheros atendiendo a su finalidad: archivos cifrados (.pgp o .asc), archivos de firma (.sig o .asc) y archivos de claves (.asc). Atendiendo a su forma, los tipos de ficheros son dos: los archivos de armadura ASCII y los archivos MIME/PGP. ¿Cuál es la diferencia entre ellos? Las armaduras ASCII son ficheros .asc de caracteres ASCII (como su propio nombre indica) que son perfectamente legibles -que no comprensibles- y pueden ser fácilmente tratados con un editor de texto plano, son cómodos para trabajar en web con ellos... los ficheros MIME/PGP, por contra, son caracteres que usan la codificación MIME y que NO pueden ser leídos con un editor de texto plano (según qué editor se use, no veremos nada o veremos símbolos extraños).

> *Para los que no sepan qué es ASCII (American Standard Code for Information Interchange), se trata de un estándar de representación numérica de caracteres. El ASCII estándar contiene 128 símbolos, y el extendido 256. Su uso ya no es muy habitual, pues ha sido progresivamente sustituido por UNICODE. Para conocer más sobre la tabla ASCII:*
> *http://www.asciitable.com/*

Los archivos cifrados y de firma pueden adoptar cualquiera de las dos formas , mientras que los archivos de claves siempre serán armaduras ASCII. Mi recomendación, para cualquier tipo de fichero que deseemos generar, es usar armaduras ASCII. Vamos a ver la pinta que tienen... para ello abrimos con un editor de texto (por ejemplo, el bloc de notas) la clave previamente exportada. Veremos una gran cantidad de letras ASCII y unas etiquetas de inicio y de final que delimitan ambas partes de la clave.

-----BEGIN PGP PRIVATE KEY BLOCK-----
Version: {versión del software}

{parte PRIVADA de la clave}
-----END PGP PRIVATE KEY BLOCK-----
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: {versión del software}

{parte PÚBLICA de la clave}
-----END PGP PUBLIC KEY BLOCK-----

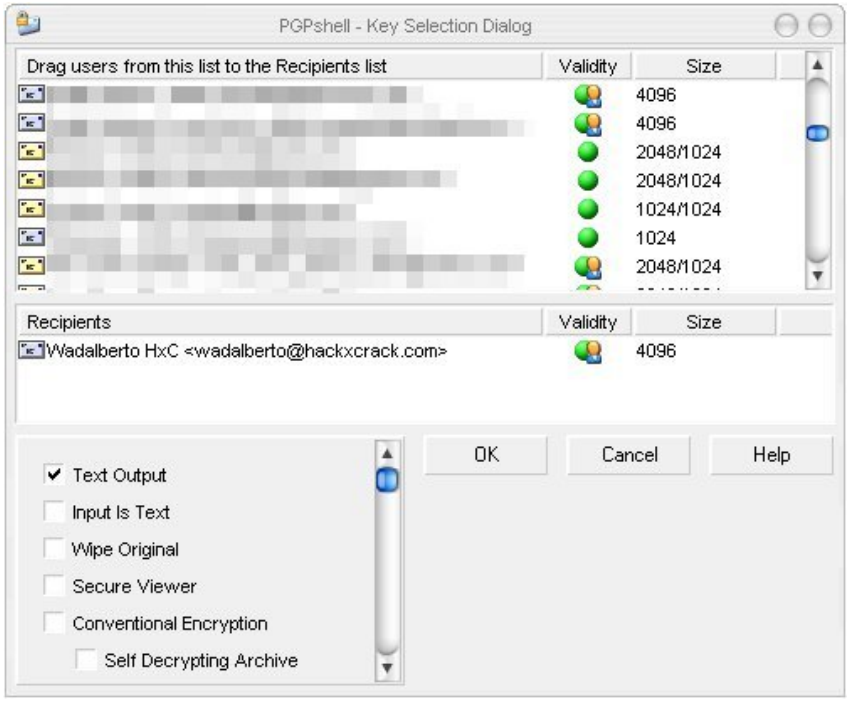
Como vemos, las cabeceras indican cuál es el contenido de la armadura ASCII (clave privada y pública respectivamente, pero podrían ser otros que más adelante veremos), y la línea “Version” indica qué software hemos usado para generar esa armadura (en vuestro caso será PGP 8.1.0). Los caracteres ASCII representan el contenido de la armadura, bien sea la clave como en este caso, un mensaje cifrado o una firma.

El trabajar con armaduras ASCII en lugar de ficheros MIME/PGP nos da una gran flexibilidad al poder tratar el contenido como texto plano. Si habéis visto alguna vez en una página web que hay un enlace a una clave PGP pública (como por ejemplo en mi web), siempre os encontraréis con el archivo .asc que puede ser visualizado correctamente mediante vuestro navegador web así como guardado de forma directa. Para las claves siempre es así, pero por ejemplo si quisiéramos publicar en un foro un fichero cifrado, sería mucho más cómodo publicar como texto la armadura ASCII que andar adjuntando el fichero MIME/PGP (además, no todos los foros soportan la opción de adjuntar ficheros).

Cifrado de archivos

Bueno, por fin hemos llegado al punto que todos estaban esperando: vamos a cifrar y descifrar archivos usando PGP. Lo primero que necesitamos es un fichero que actúe como “texto en claro” (como dijimos anteriormente, llamamos texto en claro en un criptosistema a cualquier elemento que no esté cifrado... no obstante, cualquier fichero del ordenador está compuesto por ceros y unos, por lo que en realidad no es tan complicada la abstracción de imaginarlo como un texto ;-). Podemos seleccionar un fichero cualquiera, pero para que coincida con el ejemplo, crearemos un fichero llamado texto.txt. En su interior podéis escribir lo que queráis.

Ahora pulsamos con el botón derecho sobre el fichero y seleccionamos el menú “PGP” para después seleccionar “Encrypt”. Nos encontramos con una pantalla donde podemos seleccionar (arrastrando y soltando) la o las claves que podrán descifrar el fichero cifrado. Además, tenemos a nuestra disposición las siguientes opciones:



Text Output: Al marcar esta opción, estamos indicando al programa que deseamos que genere un fichero de armadura ASCII .asc en lugar de un fichero MIME/PGP .pgp. Yo, por comodidad, recomiendo usar este tipo de ficheros cifrados, pero si preferís usar MIME/PGP, da absolutamente igual.

Input Is Texto: Mediante esta opción estamos indicando a PGP que debe tratar la entrada como si de texto se tratara... esto es algo que por el momento no vamos a usar, aunque cuando tratemos el tema de cifrado de correo electrónico veremos que es bastante importante. Por el momento olvidad esta opción.

Wipe Original: Tras cifrar el fichero, borra de forma segura el original. Más adelante hablaremos de esta opción de PGP con más detalle.

> *¿Creías que con borrar un fichero era suficiente? No, claro... pero borrarlo y vaciar la papelera de reciclaje TAMPOCO. Las técnicas y, sobre todo, los fundamentos teóricos sobre el borrado seguro de datos son un tema muy interesante pero complejo... daría para un artículo entero e independiente. Quién sabe... ;-)*

Secure Viewer: Si eres un fanático de la criptografía (o un paranoico de tres pares de narices) esta opción te encantará (eso sí, únicamente sirve para ficheros de texto). El uso de esta opción genera dos efectos en el archivo... pero que se manifestarán al descifrarlo. El primero de ellos es un aviso “eyes only” que os recordará a las películas de James Bond... el aviso es el siguiente:

> *The message you are decrypting is for your eyes only. It is recommended that this message only be read under the most secure circumstances.*

Que traducido vendría a decir algo como:

> *El mensaje que estás descifrando es sólo para tus ojos. Se recomienda que este mensaje sea visto únicamente bajo las circunstancias más seguras.*

De entrada acongoja un poco, ¿verdad? Pues veréis lo que viene ahora... Si pulsamos aceptar nos encontraremos con el "Secure Viewer" de PGP: un visor que no solo es seguro en cuanto a su forma de almacenar los datos en memoria, sino que además es seguro en cuanto a su visualización en pantalla. Si usáis la opción "Use TEMPEST Attack Prevention Font" (que está marcada por defecto), el texto se visualizará con una fuente segura contra los ataques TEMPEST.

> *Y... ¿qué es TEMPEST? Es una técnica de espionaje que ralla la ciencia-ficción, pero que os aseguro que es real. Si alguno de vosotros ha leído la genial novela "Criptonomicón" de Neal Stephenson, esta técnica la conocerá con el nombre de Phreaking Van Eck.*
> *Todos sabemos que los monitores funcionan mediante aplicación de corrientes eléctricas (en el caso de monitores CRT, mediante el barrido de electrones; en el caso de TFT, mediante aplicación de diferencias de potencial a los pixels), y que las corrientes eléctricas generan campos eléctricos...*
> *Pues el ataque TEMPEST consiste en detectar de forma remota los campos eléctricos generados por un monitor (podéis imaginar que son muchísimos) e interpretarlos para recomponer, también de forma remota, la imagen que generó esos campos. Casi nada, vamos... Si queréis saber más sobre TEMPEST:*
> *http://www.eskimo.com/~joelm/tempest.html*

Conventional Encryption: Al activar esta opción, el fichero se cifra únicamente con una clave simétrica (de las que hablamos en las opciones de PGP) con la clave que nosotros introduzcamos. Para entender la diferencia con el cifrado estándar de PGP hay que esperar un poquito a que lo veamos.

Self Decrypting Archive: Genera un fichero con cifrado simétrico que no necesita de PGP para ser descifrado. Para entenderlo mejor, un SDA sería a PGP como un EXE autoextraíble a WinZip. También se pueden crear SDA's directamente desde el menú PGP sobre el fichero con la opción "Create SDA".

Bien, ya sabemos cómo cifrar archivos con PGP... pero queda lo más interesante (al menos si eres una mente inquieta): saber cómo cifra PGP.

Lo primero que debemos tener en cuenta es que el coste computacional (en potencia y, por tanto, en tiempo) de cifrar algo a una clave asimétrica (más si hablamos de claves grandes, de 2048 bits o más) es MUY grande comparado a, por ejemplo, el cifrado simétrico o la generación de hashes. Pero las claves asimétricas son muy poderosas, podríamos decir que son virtualmente irrompibles (romperlas con las actuales técnicas requeriría más tiempo de lo que lleva existiendo el Universo...).

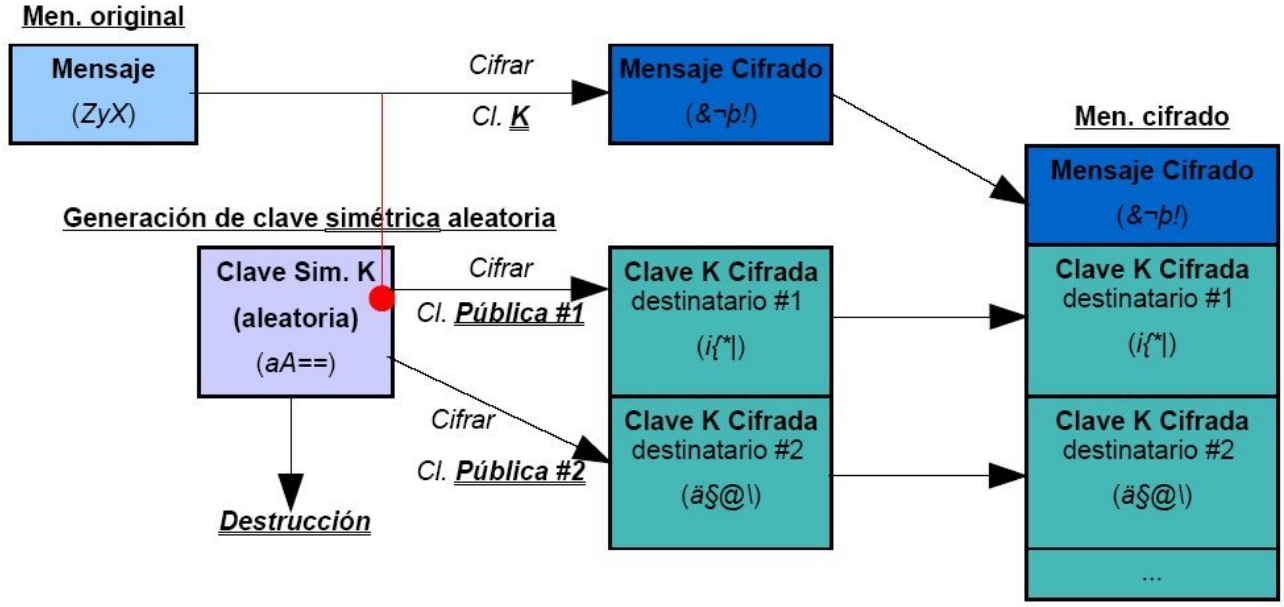
En el otro lado del cuadrilátero tenemos el cifrado simétrico, que es muy rápido, pero tiene una pega importante: al ser única la clave, a partir de varios mensajes distintos cifrados a la misma clave, se pueden realizar ataques criptoanalíticos (el criptoanálisis es la rama opuesta de la criptografía, que se encarga de descifrar criptogramas. Ambas ramas conforman lo que denominamos criptología) de muchos tipos: fuerza bruta, ataques estadísticos, probabilísticos... y son bastante más fáciles de romper que los asimétricos (lo cual no quiere decir, ni mucho menos, que sea fácil).

> *Como curiosidad histórica: la máquina Enigma usada por los Alemanes durante la Segunda Guerra Mundial era un aparato criptográfico simétrico tremendamente poderoso para ser mecánico y para la época en que existió. Pero, como ya dijimos antes, una cadena es tan dura como el más débil de sus eslabones, y aunque la Enigma aumentó su complejidad durante la Guerra (con nuevos rotores, más permutaciones del teclado...), el procedimiento exigía que se cifrara por duplicado la clave del día al inicio del mensaje. Este gravísimo error permitió a la gente de Bletchley Park romper el cifrado Enigma una y otra vez, dado que ese pequeño detalle reducía la complejidad del criptoanálisis del problema enormemente.*

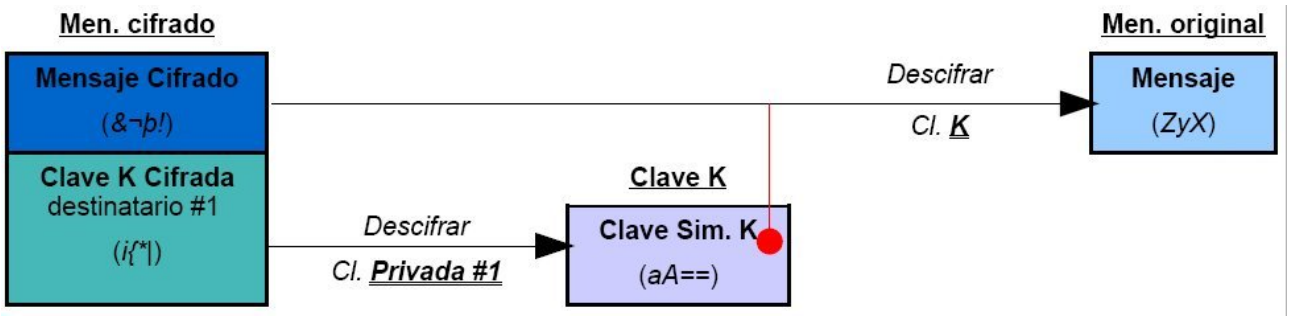
Ya hemos dicho que el principal problema de un criptosistema simétrico es que el cifrado repetido de mensajes a una misma clave facilita la ruptura de la misma... por tanto, ¿porqué no usar una clave simétrica para cada mensaje? Claro, sería un engorro cambiar la clave cada vez... porque necesitaríamos un canal seguro para transmitirla y sería la vuelta al principio, ¿verdad? Pues NO.

Una clave simétrica es muchísimo más pequeña (256 bits como máximo) que cualquier mensaje que vayamos a cifrar, por pequeño que sea éste. Así pues, cifrar la clave simétrica con una clave asimétrica no requiere demasiado tiempo... pues eso es exactamente lo que hace PGP (y otros estándares cifrados como SSH o SSL).

Al cifrar un archivo, se genera una clave simétrica aleatoria (nota para los puristas: pseudoaleatoria) que llamaremos clave de sesión con la que se cifra el mensaje. Posteriormente se cifra la clave de sesión con la clave pública de cada uno de los destinatarios del mensaje. El último paso es destruir la clave de sesión y empaquetar todos los datos en una armadura ASCII (o bien un fichero MIME/PGP). Mediante este mecanismo tan sencillo e ingenioso logramos un cifrado rápido dado que el mensaje (la parte de mayor tamaño) es cifrado a una clave simétrica, a la vez que muy seguro porque la clave simétrica es usada una sola vez y se transmite mediante cifrado asimétrico.



Descifrar archivos es mucho más sencillo que cifrarlos. Podemos o bien seleccionar la opción “Decrypt & Verify” (la verificación corresponde a la parte de firmas, que veremos a continuación) del menú contextual de PGP, o bien directamente hacer doble click sobre el fichero. Solamente puede darse el caso especial del que el fichero haya sido cifrado con la opción “Secure Viewer”, en cuyo caso saldrá el diálogo de advertencia (que deberemos aceptar o cancelar) y posteriormente, si aceptamos, el visor seguro de datos con el mensaje en cuestión.



¿Cómo funciona el descifrado de datos en PGP? Una vez que sabemos cómo funciona el cifrado, el paso contrario es trivial. Cuando desciframos un fichero lo primero que ocurre es que mediante nuestra clave privada desciframos la parte que contiene la clave de sesión del mensaje y que fue cifrada a nuestra clave pública (si además de la nuestra fue cifrada a otras, ignoraremos los demás fragmentos que contienen la clave de sesión cifrada a esas otras claves públicas), para a continuación usar esa clave de sesión para descifrar el mensaje original.

Firma PGP

Otra de las aplicaciones más importantes del sistema PGP es la firma digital. Todo sistema de firma digital debe cumplir unas características:

1) Integridad de la información: En caso de que la información se vea alterada tras su firma, ésta deja de ser válida.
2) Autenticidad: Asegura que el solamente el dueño de la clave privada ha sido capaz de generar la firma.
3) No repudio: El usuario que generó la firma no puede negar haberlo hecho.

Ahora llega otro de nuestros ejercicios de imaginación... ;-)

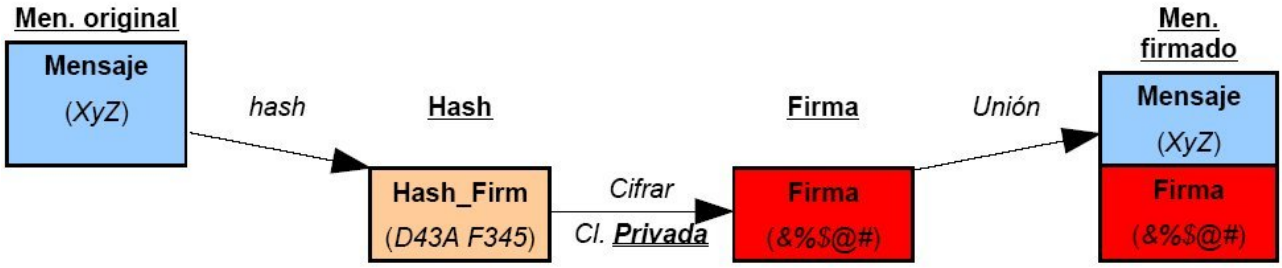
Imaginemos que queremos idear, con lo que sabemos, una forma de implementar una firma digital que cumpla las características anteriormente citadas. Seguramente lo primero que se os ocurra a la mayoría sea realizar un hash del fichero a firmar y pasarle la cadena generada junto al fichero. La primera de las características se cumple, pues como sabemos, si modificamos cualquier dato del fichero, el hash dejará de coincidir. Pero la segunda no se cumple (y, por consiguiente, la tercera tampoco), dado que cualquiera puede generar un hash de un fichero dado... y también cualquiera podría interceptar un envío con un fichero y su hash, modificar el fichero, recalcular el hash y reenviarlo, todo ello sin que el destinatario tenga ninguna forma de saber si esta modificación ha ocurrido.

Demos una vuelta más de tuerca al concepto anterior para aumentar la seguridad del sistema. Calculemos el hash del fichero y luego cifremos este hash a la clave pública del destinatario. Bien, este método tampoco resultaría válido, porque cualquier persona puede cifrar a una clave pública, así que únicamente lograríamos complicar la vida un poco a un posible atacante, que tendría que modificar el fichero, recalcular el hash y codificar el mismo a la clave pública del destinatario. Aún así, hemos introducido un elemento interesante en nuestra particular ecuación: el atacante no podrá visualizar la información que quiere modificar, en este caso el hash.

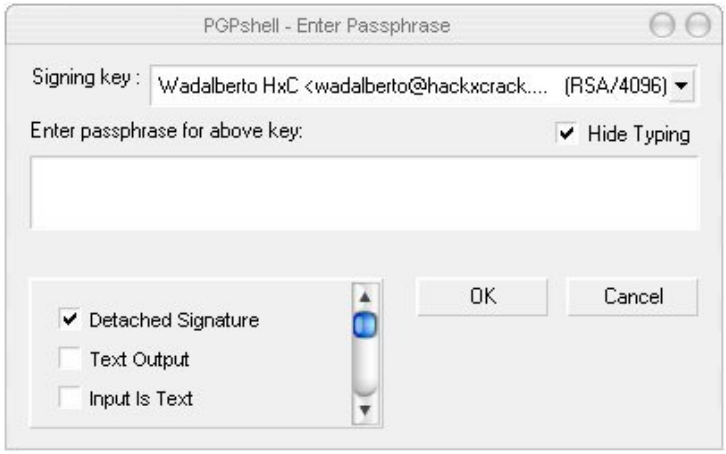
Si recordáis los tiempos del instituto, esos problemas donde había un “paso mágico” que o se te ocurría o no había forma de continuar... pues aquí es donde ocurre nuestro particular “paso mágico”. Como dijimos casi al principio, una clave asimétrica se compone de dos claves COMPLEMENTARIAS. Esto significa que lo que cifremos con una, puede ser descifrado con la otra y viceversa. El método normal de uso de estas claves en PGP es cifrar algo a una clave pública, de forma que el usuario de la clave privada y solamente él pueda descifrar esa información, pero... ¿y si lo que queremos es, como en este caso, realizar un cifrado que únicamente nosotros podamos realizar y que cualquier persona puede comprobar? Es justo lo contrario... por lo que deberíamos realizar justamente el cifrado contrario. ¡Claro!

Esa es la idea: cifrar algo a nuestra clave privada, de forma que cualquiera que tenga nuestra clave pública (la cual se debería poder descargar de un servidor de claves) pueda descifrar ese mensaje. Ahora llevemos ese concepto al problema concreto que nos planteamos. Imaginemos que generamos un hash del fichero a firmar, y ese hash lo ciframos a nuestra clave privada. Pues nos encontramos con una firma digital PGP.

Proceso de firma:



Para fimar un fichero, debemos pulsar con el botón derecho sobre el fichero y seleccionar la opción “Sign”. Adicionalmente podemos seleccionar “Encrypt & Sign” para cifrar y firmar. Al seleccionar la opción de firma, nos encontraremos con una ventana con las siguientes opciones:



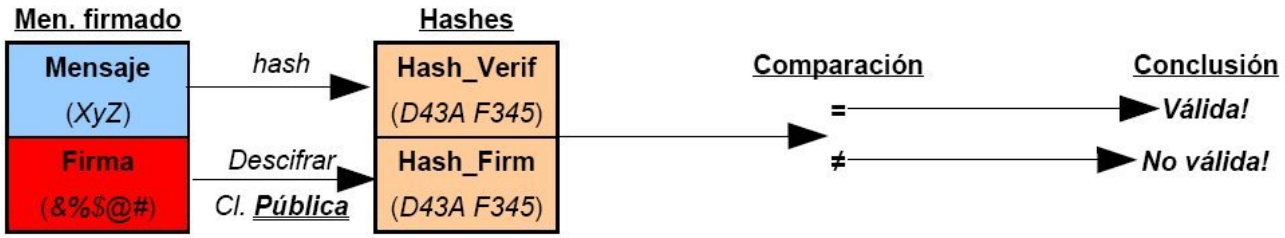
Detached Signature: Al seleccionar esta opción (que está activada por defecto), la firma estará separada del fichero, generando un fichero independiente cuyo nombre será el nombre completo del archivo firmado (incluyendo su extensión) y la extensión .sig (independientemente de que sea una firma en la forma de armadura ASCII o PGP/MIME). Por ejemplo, prueba.txt generaría la firma prueba.txt.sig.

En caso de no seleccionar esta opción, se generará un fichero de extensión .asc (para armadura ASCII) ó .pgp (para PGP/MIME) que contendrá el fichero y su firma. El sistema de generación del nombre será igual, y prueba.txt pasaría a ser o bien prueba.txt.asc (para armadura ASCII) o bien prueba.txt.pgp (para PGP/MIME).

Text Output: Al seleccionar esta opción, estamos indicando que deseamos generar la firma en la forma de armadura ASCII.

Input Is Text: Seleccionando esta opción estamos indicando al software que la entrada debe ser tratada cono texto.

Cuando el destinatario reciba el fichero y su hash, realizará dos operaciones: por un lado, obtendrá por su propia cuenta el hash del fichero (obviamente, utilizando el mismo algoritmo que usamos nosotros); y por el otro, utilizará nuestra clave pública para descifrar el hash que nosotros generamos. Solamente queda comparar ambos hashes: si son idénticos, la firma es válida y podemos garantizar que la información no ha sido alterada y que la generó el firmante.

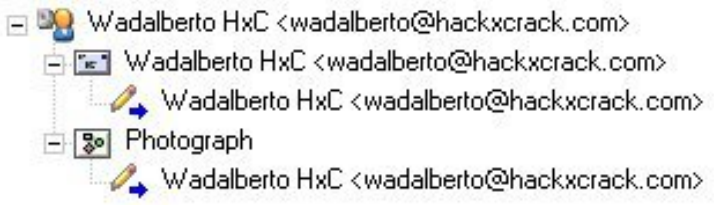


Para verificar una firma (sea el fichero cifrado o no) debemos pulsar sobre él con el botón derecho y seleccionar la opción “Decrypt & Verify”.

Imaginemos que estamos en la situación del atacante. En esta ocasión si puede descifrar el hash, pues puede obtener nuestra clave pública de cualquier servidor de claves. También puede alterar el fichero y generar su propio hash... pero jamás podrá cifrar ese hash a nuestra clave privada. Simple y efectivo.

El anillo de confianza

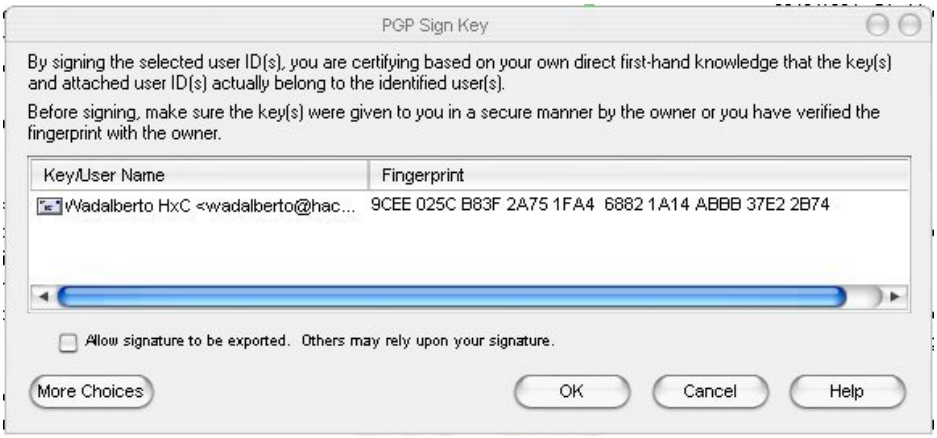
La firma digital se puede aplicar a cualquier tipo de información que maneje el ordenador, no únicamente a ficheros. Por ejemplo, podemos firmar... claves públicas. Si os fijáis en el árbol de vuestra propia clave, vuestra propia clave pública se encuentra por defecto firmada por vuestra clave privada desde el momento de su generación. ¿Y cuál es la utilidad de firmar claves públicas?



Como ya hemos visto, una firma únicamente puede generarla el dueño de la parte privada de la clave. Así pues, cuando firmamos una clave pública (es decir, ciframos a nuestra clave privada el hash de esa clave pública) estamos dando testimonio de que esa clave es de confianza, es decir, que pertenece a quien dice pertenecer. Así, si yo he firmado la clave de Wadalberto HxC porque le conozco y confío en él, si tú me conoces y confías en mí (bueno, y en mi palabra :-P), puedes considerar la clave de Wadalberto HxC como buena. Así se genera un “anillo de confianza”.

> *Hay gente que se toma esto muy en serio. Por ejemplo, los desarrolladores de Debian (una de las distribuciones de GNU/Linux más famosas) conforman un enorme anillo de confianza, y para entrar en él, el proceso de firma de claves públicas se realiza entre las dos personas implicadas... en persona.*
>
> *En el VI Congreso de Software Libre de HispaLinux, celebrado en Septiembre de 2003, hubo una charla de GnuPG en la que hubo un intercambio de claves donde los participantes intercambiaban sus KeyID y fingerprint... y hasta dejaron sus DNI's encima de una mesa para que todo el mundo pudiera comprobar las fotografías... no os asustéis, yo no llego a ese extremo... :-P*

Existen principalmente dos tipos de firma sobre clave pública en PGP: la exportable y la no exportable. La primera de ellas será exportada al exportar la clave (bien a un fichero o bien a un servidor de claves), mientras que la segunda únicamente tiene validez dentro de nuestro anillo de claves y no será exportada en ningún caso. Para que el concepto de anillo de confianza funcione, se deben usar firmas exportables y, tras la firma, exportar la clave a los servidores de claves apropiados, de forma que los cambios se efectúen en éstos y cualquier persona que descargue o actualice la clave desde el servidor obtenga los cambios efectuados sobre ella.



Para firmar una clave pública con PGP debemos seleccionarla y pulsar con el botón derecho sobre ella para después elegir "Sign" (para firmar únicamente una parte de la clave el proceso es el mismo pero seleccionando el elemento en cuestión). Veremos una pantalla donde se nos muestra un mensaje en el que nos advierten de las implicaciones de firmar una clave pública, el nombre de la clave y el fingerprint de la misma. Hay también una casilla en la que podemos marcar la opción "Allow signature to be exported. Others may rely upon my signature." que activa la opción de firma exportable.

> *Pulsando en el botón "More Choices" nos encontramos con las opciones avanzadas de firma de PGP. En este menú podremos elegir dos tipos de firmas PGP peculiares y no muy usadas: el "Meta-Introducer Non-Exportable" y el "Trusted Introducer Exportable", así como la opción de generar cualquiera de los cuatro tipos de firma con una fecha de expiración. Estos dos tipos de firmas son usadas para describir una confianza más profunda, detallada en su nivel de confianza y con la posibilidad de restringir el dominio donde exista el correo de la clave. No daré más detalles de estas opciones porque son usaras muy raramente (de hecho no las he visto más que un par de veces).*

Al aceptar este diálogo pasaremos a otro donde debemos seleccionar la clave privada que queremos usar para firmar esta clave pública (en caso de disponer de más de una clave privada) e introducir su passphrase para confirmar la opción.

Cuando queremos comprobar que una clave pública mantiene todas sus firmas y no han existido revocaciones (de lo que hablaremos a continuación) debemos sincronizar la clave de nuestro anillo con la del servidor de claves. Para ello pulsaremos sobre la clave con el botón derecho y seleccionaremos la opción "Reverify Signatures".

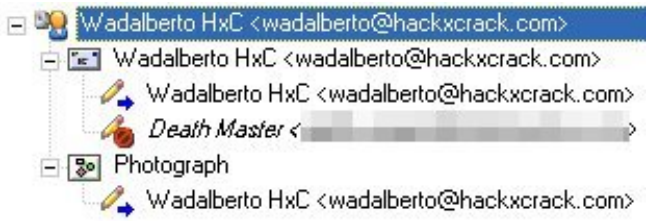
Es importante tener en cuenta que para poder establecer una confianza en una clave pública (ya vimos cómo se hacía esto cuando hablamos de la clave PGP) es imprescindible haber firmado previamente la misma. Por ello, es útil utilizar las firmas no exportables para poder establecer nuestro sistema de confianza en nuestro anillo local de claves.

Revocaciones

Las revocaciones son un mecanismo ideado para poder dar marcha atrás en algunos procesos críticos de PGP, como por ejemplo la publicación de claves o la firma de claves públicas.

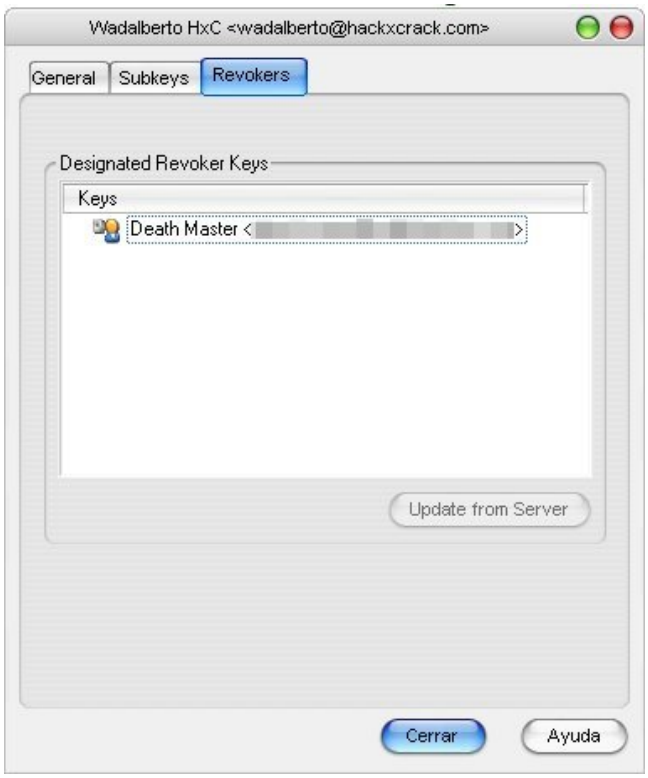
Imaginemos que nos generamos una nueva clave y deseamos eliminar la vieja. Si nuestra clave ha tenido bastante difusión y está publicada en servidores de claves nos encontraremos con el problema de que cualquier persona, por error, puede bajarse la clave antigua y utilizarla... y quizá ni siquiera el correo electrónico de la clave sea válido ya. Para evitar estas situaciones, podemos revocar de forma completa nuestra clave y después actualizarla en los servidores de claves. De esta forma, la próxima vez que alguien actualice nuestra clave pública, nuestra clave en su anillo local pasará también a estar revocada y NO podrá volver a cifrar a esa clave.

El concepto de revocaciones puede ser también aplicado a las firmas de claves públicas. Imaginemos que hemos firmado por error una clave pública de forma exportable y hemos actualizado los servidores de claves con ella (hombre, hacer todo eso por error... ya es despiste, sí... :-P) o que una clave que habíamos firmado resulta ser falsa, falta de confianza... o cualquier cosa por el estilo.
Podemos revocar nuestras firmas sobre otra clave de forma que quede constancia que nosotros explícitamente hemos eliminado la validez de esa firma. Al igual que en la revocación de claves completas, al actualizar la clave (y en este caso además, al sincronizar las firmas con la opción "Reverify Signatures") se actualiza el anillo local de claves con las revocaciones correspondientes.



Existe un tipo especial de firmas exportables que tienen como propiedad especial la imposibilidad total de ser revocadas. Una vez generada una firma de este tipo y actualizada la clave en el servidor pertinente, es absolutamente imposible revocar esa firma. Esta opción NO está implementada en PGP, pero por ejemplo GnuPG sí la implementa.

Existe una última opción de las revocaciones por la cual podemos añadir revocadores a nuestra clave. Añadir un revocador (representado por otra clave pública) a nuestra clave significa que estamos otorgando al propietario de esa clave permiso para revocar nuestras firmas. Podemos imaginar un ejemplo en el que esta característica sería útil: imaginemos un equipo de trabajo en el que hay un jefe y varios empleados y cuyo trabajo se coordina con archivos cifrados y un sistema de claves en anillo de confianza. Podría ser una obligación de cada uno de los empleados el añadir a su jefe como revocador, de forma que las firmas de cada empleado pudieran ser revocadas o bien por el mismo usuario o bien por su jefe. Esta opción realmente tampoco es muy usada.



Para añadir revocadores a nuestra clave pulsaremos sobre ella con el botón derecho y elegiremos "Add" seguido de "Revoker". Al consultar las propiedades de la clave veremos una nueva pestaña de título "Revokers" donde podremos consultar los revocadores definidos en nuestra clave.

Claves compartidas

Esta opción es una de las más avanzadas e interesantes de PGP, pero no se usa prácticamente. Gracias a la opción de definir claves compartidas, podemos dividir una clave completa (con parte pública y privada) en varias "subclaves". Cada una de esas subclaves tendrá su propio passphrase y un valor numérico asociado que define su importancia, y podremos igualmente definir el valor total necesario para realizar cualquier acción con la clave.

Para dividir una clave pulsaremos en ella con el botón derecho y seleccionaremos “Share Split”. En la pantalla que aparecerá a continuación podremos definir cada usuario y su valor, así como el valor total necesario para llevar a cabo acciones. Al pulsar en el botón “Split Key” el programa nos pedirá que le indiquemos un directorio donde almacenar los fragmentos de la clave. A partir de ese momento, cualquier acción que deseemos realizar con esa clave estará precedida de un diálogo de reconstrucción de la clave donde se irán solicitando los archivos y passphrases pertinentes.

Haciendo un símil, podemos considerar esta opción como las famosas dobles llaves que tenían que activar simultáneamente los operarios de silos atómicos para realizar la orden de lanzamiento.

Borrado seguro

Ha llegado el momento de hablar de una de las opciones más útiles de PGP y que, de hecho, no está relacionada con la criptografía ni con las claves... pero que no podéis perderos y no quería dejar de comentar. :-)

Como ya dije antes, no basta con borrar un fichero para que éste desaparezca del disco duro. Podemos imaginarnos el disco duro como un enorme libro que tiene un índice y unos contenidos. Cuando escribimos un algo (un fichero), estamos escribiendo algo en alguna página e introduciendo en el índice una entrada donde indicamos la posición de esa información para saber que está ocupada. Al borrar un fichero, el sistema operativo en realidad lo que hace es borrar el índice, decir que ese espacio está libre para ser usado si es necesario... pero no borra la información realmente.

Seguramente ya habréis pensado la mayoría que bastaría entonces con sobreescribir la información real para eliminar los datos. Sí y no. Sí porque así realmente desaparecería la información, pero no por varios motivos: en primer lugar, un usuario desde el sistema operativo normalmente no tiene posibilidad de controlar el disco hasta tal punto como para escribir los clusters exactos donde se alojaba la información; y en segundo lugar porque aunque sobreescribamos la información, ésta podría ser recuperada bajo ciertas circunstancias.

> *La mayoría sabréis que un disco duro es un dispositivo magnético y que la información se guarda como bits magnetizados. Estos bits generan millones de mini-campos magnéticos. Si habéis estudiado física eléctrica, sabréis que al aplicar un mismo campo (por ejemplo, si queremos escribir un 1) a dos zonas donde los campos no son los mismos (por ejemplo, dos bits donde había 1 y 0 respectivamente), los campos resultantes no son idénticos. Dado que el ordenador trabaja con unos márgenes de error (podéis verlo en los voltajes del procesador, por ejemplo), ambos se interpretarían como un uno, pero con unos aparatos de medida suficientemente precisos, podríamos averiguar qué había escrito ANTES en esa zona del disco duro, independientemente de lo que haya escrito ahora.*

Para evitar que esa información pudiera ser recuperada se deben usar herramientas de borrado seguro de datos. Hay varios métodos para esto: sobreescritura ISAAC, DoD 5220-22.M del Departamento de Defensa de EEUU (basado en ISAAC), método Gutmann... pero no es la finalidad de este artículo el profundizar sobre este tema.

Sí diremos que PGP incorpora una herramienta de borrado seguro de datos llamada “Wipe” que está basada en el método DoD 5220-22.M y que permite una reescritura segura de los datos mediante un número de pasadas configurable desde las opciones de PGP (por defecto 3, y eso es mucho más que suficiente).

Para utilizar esta herramienta simplemente debemos seleccionar el/los fichero/s a borrar de forma segura y pulsar con el botón derecho en ellos, seleccionar el menú “PGP” y la opción “Wipe”. Un diálogo de confirmación aparecerá antes de llevar a cabo la acción (este diálogo puede eliminarse... pero no lo recomiendo en ningún caso): conviene estar muy seguro antes de decir sí... porque cualquier cosa borrada con Wipe será totalmente irrecuperable.

Terminando...

Hemos terminado con el primer artículo de este Taller de Criptografía. Espero que os haya resultado interesante y que a partir de ahora incorporéis PGP a vuestros sistemas como un elemento imprescindible... ;-)

Hasta el próximo número, con GnuPG.

2- GnuPG

(Publicado en "PC Paso a Paso: Los cuadernos de Hack X Crack" número 28, 11 de Abril de 2005)

Bienvenidos de nuevo al rincón más críptico de la revista. Espero que el primer artículo os haya gustado. :-)

En este segundo artículo vamos a conocer al "hermano libre" de PGP, GnuPG, además de conocer algunas de las interfaces gráficas disponibles para el mismo. Como ya conocemos las bases del funcionamiento del sistema OpenPGP del artículo anterior, no tiene sentido repetirlas de nuevo... pero sí vamos a profundizar un poquito más. Si alguien se siente perdido con toda la teoría del sistema, puede echar mano del artículo anterior para aclararse.

Pero antes de entrar de lleno con nuestro querido GnuPG, vamos a hablar de algo que está de actualidad en el mundo de la criptografía y que, aunque os pueda parecer lo contrario, está muy relacionado con el sistema OpenPGP. Al que no le interese mucho esta parte, puede saltar tranquilamente a la parte de GnuPG, siempre que no le importe arder eternamente en las llamas del infierno reservado a los "faltos de curiosidad". }:-D

Autopsia del ataque a SHA-1

Es curioso ver que el mundo de la criptografía, donde mucha gente piensa que ya está todo inventado, está tan vivo como estuvo en los tiempos de Bletchley park y Mr. Turing. Hace un par de meses, en el artículo anterior, mientras explicaba los algoritmos hash dije:

<<SHA-1 (Secure Hash Algorithm), 1994. SHA-1 nació de manos del NIST como ampliación al SHA tradicional. Este algoritmo, pese a ser bastante más lento que MD5, es mucho más seguro, pues genera cadenas de salida de 160 bits (a partir de entradas de hasta 2^64 bits) que son mucho más resistentes a colisiones simples y fuertes.>>

Pues parece que el destino quiso quitarme la razón, porque apenas unos días después de que saliera a la venta la revista, el 15 de Febrero, salta la liebre en las comunidades dedicadas a la criptografía: Bruce Schneier ha publicado en su blog una bomba: SHA-1 ha sido roto:

http://www.schneier.com/blog/archives/2005/02/sha1_broken.html

¿Qué? ¿¿Roto?? Comienzan las elucubraciones y la gente comienza a opinar sobre las implicaciones de este hecho... pero en realidad todavía no se conocen ni los detalles, solo una somera nota de los resultados obtenidos por los investigadores Xiaoyun Wang, Yiqun Lisa Yin, y Hongbo Yu (Universidad de Shandong, China).

Si nos atenemos al algoritmo SHA-1 propiamente dicho, este ataque ha permitido su criptoanálisis en 2^69 operaciones frente a las 2^80 que cabrían esperar en un algoritmo de su clase y longitud. ¿Qué significa eso realmente? Aún no se conocen los detalles y tampoco si este ataque logró obtener colisiones simples, colisiones fuertes...

Dos días después, el día 17, pudimos tener acceso a un documento original con los resultados (fechado el día 13 de Febrero), pero en él no se trata el ataque a SHA-1, sino únicamente uno al SHA original y otro a una versión simplificada de SHA-1.

http://theory.csail.mit.edu/~yiqun/shanote.pdf

Al día siguiente, el día 18, el mismo Bruce Schneier publica en su blog más detalles sobre el criptoanálisis aplicado a SHA-1:

http://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html

En los foros, listas y comunidades de seguridad se sigue hablando tímidamente del tema, pero realmente poca gente comprende las implicaciones reales, a corto y medio plazo, que ha tenido este ataque sobre el que es, hoy por hoy, el principal algoritmo hash en la criptografía moderna.

Antes de nada, deberíamos hacer un pequeño repaso a las características de un algoritmo hash para entender bien la explicación:

> *Unidireccional: Una función hash es irreversible, y dada una salida de la misma, es computacionalmente imposible recomponer la entrada que la generó.*
>
> *Compresión: Dado un tamaño cualquiera de entrada para la función hash, la salida de la misma será de una longitud fija.*
>
> *Difusión: La salida de la función hash es un resumen complejo de la totalidad de la entrada, es decir, se usan TODOS los bits de la misma.*
>
> *Resistencia a las colisiones simples: Esta propiedad nos indica que dado un valor de entrada de la función hash, es computacionalmente imposible encontrar otra entrada que genere el mismo valor de salida.*
>
> *Resistencia a las colisiones fuertes: Esta propiedad nos indica que es computacionalmente muy difícil encontrar dos valores de entrada que generen un mismo valor de salida.*

Como podemos ver, tenemos tres características "fijas" que son implementadas en el propio algoritmo: unidireccionalidad (también conocida como resistencia a la preimagen), compresión y difusión. Las debilidades, por tanto, se nos pueden presentar en la resistencia a las colisiones simples (también conocida como resistencia a la segunda preimagen) y en la resistencia a las colisiones fuertes (también conocida como resistencia a la colisión).

Debemos tener en cuenta un par de cosas cuando pensemos en las colisiones de un sistema de hash:

- El conjunto de los posibles valores de entrada tiene cardinal infinito, puesto que en principio podemos aplicar el algoritmo de hash a cualquier entrada. Esto no se cumple para SHA-1 en concreto, dado que está limitado a entradas de tamaño menor o igual a 2^64 bits. Por tanto para SHA-1 concretamente nos encontramos con 2^(2^64) posibles entradas, es decir, 2^(1.8*10^19). No he calculado el valor concreto (más que nada porque Kcalc se desborda :-D) pero es una auténtica burrada y para nosotros eso nos vale como infinito...

- El conjunto de posibles valores de salida tiene un cardinal finito que viene definido por 2^n donde n es el número de bits de la salida generada, es decir, por la propiedad de compresión. En SHA-1 tenemos 2^160 posibles salidas (1.46*10^48).

De estas dos propiedades deducimos que es imposible que toda entrada tenga una salida diferente, puesto que el cardinal del conjunto de entrada es mayor que el de salida. Así pues, si generamos (2^160)+1 mensajes, será inevitable que al menos dos de ellos posean el mismo valor de hash.

Hasta ahí la teoría de las colisiones. En la práctica encontramos dos tipos de colisiones:

- Colisiones simples: Si dado un mensaje de entrada concreto, somos capaces de encontrar otro de forma que ambos posean el mismo valor de hash, nos encontramos ante una colisión simple.

- Colisiones fuertes: Si somos capaces de encontrar dos mensajes de entrada con un mismo valor de hash, nos encontramos ante una colisión fuerte.

Y es la hora de hablar de colisiones y cumpleaños (y que conste que aún no he perdido la cabeza).

Vamos a trasladar las colisiones simples a un ejemplo que sea más fácil de digerir... imaginad que un hombre entra en una fiesta y desea saber las posibilidades de que otra persona cumpla años el mismo día que él. Para los que no tengan la combinatoria muy al día o no les apetezca pensar mucho, las posibilidades son 1-(364/365)^n. Es decir, para que las posibilidades sean superiores al 50% es necesario que haya 253 personas en esa fiesta.

Ahora pensemos en las colisiones fuertes... imaginemos ahora que queremos estudiar las probabilidades de que dos personas cuales quiera cumplan años el mismo día. Las posibilidades ahora son 1-[365!/(365^n*(365-1)!)], osea que ahora únicamente son necesarias 23 personas o más para que las posibilidades sean superiores al 50%.

A este modelo matemático se le conoce como "ataque del cumpleaños".

¿Y qué significa ésto en la práctica? Significa que si quisiéramos lanzar un ataque de fuerza bruta contra un algoritmo de hash de n bits, necesitaríamos realizar teóricamente 2^n operaciones para encontrar una colisión simple, mientras que para encontrar una colisión fuerte necesitaríamos realizar únicamente 2^(n/2). En el caso particular de SHA-1 nos encontramos con que en teoría serían necesarias 2^160 operaciones para obtener una colisión simple y 2^80 para obtener una colisión fuerte.

Supongo que ahora ya se entienden un poco mejor las implicaciones del ataque a SHA-1. Como hemos dicho, la teoría nos dice que para obtener una colisión fuerte necesitaríamos 2^80 operaciones, y mediante el ataque del que estamos hablando se obtiene en solamente 2^69 operaciones. Puede parecer que no es mucha, pero dado que cada bit en el exponente duplica la complejidad (como ya dije en el primer artículo), la diferencia entre 2^69 y 2^80 es enorme, unas dos mil veces más sencillo (concretamente 2^11, 2048).

Las implicaciones prácticas de esto a corto plazo no son tan graves como mucha gente se aventuró a anunciar. El sistema OpenPGP no ha dejado de ser seguro, pues aunque SHA-1 sea dos mil veces más sencillo de comprometer, sigue siendo una tarea muy compleja.

Además, para poder falsificar un mensaje sería necesario encontrar una colisión simple, y eso sigue requiriendo 2^160 operaciones, lo cual es a todas luces seguro. Lo que sí sería posible es crear dos mensajes cuya firma sea la misma, y aunque no es lo mismo, sí es algo a tener en cuenta.

A largo plazo las implicaciones son peores, dado que la potencia de cálculo aumenta constantemente, y lo que hoy es complicado, mañana es sencillo... el precalcular colisiones fuertes en SHA-1 ya solo es cuestión de el tiempo que se quiera invertir en ello.

No obstante, el alarmismo que ha cundido es injustificado, pues hay muchos sistemas que han sido rotos, como MD5, SHA-0, DES y otros tantos, que siguen siendo usados. De hecho, como comenté en el primer artículo, MD5 sigue siendo un estándar de facto, y hace mucho tiempo que fue comprometido en una forma similar a la que acaba de serlo SHA-1. La diferencia está en que ahora mismo se confía en MD5 para tareas de "baja" seguridad (como checksums de imágenes ISO) mientras que se confía en SHA-1 para aplicaciones más críticas, como firma y certificados digitales.

¿Qué camino va a tomar la comunidad criptográfica? La gente de la PGP Corporation ya han anunciado que en sucesivas versiones incluirán nuevos algoritmos, sin prisa, pero de forma segura.

Así mismo se va mirando hacia nuevos algoritmos para reponer al herido SHA-1:

- SHA-256, SHA-384, SHA-512: Versiones de 256, 384 y 512 bits respectivamente de SHA. Corren rumores de que PGP se inclinará por alguno de estos algoritmos. Podéis encontrar más información acerca de ellos en:

http://csrc.nist.gov/encryption/shs/sha256-384-512.pdf

- RIPEMD-160: Algoritmo de hash de 160 bits diseñado por Hans Dobbertin, Antoon Bosselaers, y Bart Preneel. Definitivamente no es un sustituto de SHA-1, pues también ha sufrido ataques de criptoanálisis que han permitido encontrar colisiones:

http://eprint.iacr.org/2004/199.pdf

Si es factible, no obstante, la implementación de versiones más potentes de RIPEMD, como RIPEMD-320.

- Tiger: Algoritmo de hash de 192 bits diseñado en 1996 por Ross Anderson y Eli Biham. Está especialmente optimizada para ser utilizada en procesadores de 64 bits, y es vista con buenos ojos por los expertos. Podéis encontrar más información sobre Tiger en:

http://www.cs.technion.ac.il/~biham/Reports/Tiger/

- WhirPool: Algoritmo de hash de 512 bits diseñado por Vincent Rijmen y Paulo S. L. M. Barreto. No es muy conocido, y dado que gran parte de su confiabilidad se basa en su gran longitud, resulta bastante lento computacionalmente hablando. Podéis leer más sobre WhirPool en:

http://planeta.terra.com.br/informatica/paulobarreto/WhirlpoolPage.html

En cualquier caso, el futuro no parece pasar por la convocación de un concurso al estilo AES para que la comunidad internacional lance sus propuestas y de ellas salga el nuevo estándar de hashing. Algunos expertos, entre ellos Bruce Schneier, ya lo están pidiendo.

Bien, espero que ahora tengáis más claras las implicaciones del ataque criptoanalítico sobre SHA-1. Ahora podemos meternos de lleno con GnuPG. :-)

GNU Privacy Guard: Un poco de historia

GNU Privacy Guard (al que nos referiremos como GPG) es una implementación libre del sistema OpenPGP (descrito en el RFC #2440, como ya dijimos en el artículo anterior). Ofrece compatibilidad con PGP e implementa todos los mismos algoritmos que éste menos el algoritmo de cifrado simétrico IDEA, por ser éste aún un algoritmo patentado en ciertos países. Aún así, es posible añadir la implementación del mismo al sistema hasta que sea oficialmente incluído en 2007 (cuando caduque la patente).

El sitio oficial de GPG es:

http://www.gnupg.org/

Existen muchas ventajas derivadas de que GPG sea software libre, como por ejemplo la gran cantidad de sistemas soportados (GNU/Linux, GNU/Hurd, FreeBSD, OpenBSD, NetBSD, Microsoft Windows, PocketConsole, Mac OS X, AIX, BSDI, HPUX, IRIX, MP-RAS, OSF1, OS/2, SCO UnixWare, SunOS, Solaris y USL UnixWare) o la disponibilidad total de su código fuente (y no únicamente como material de consulta, como en el caso de PGP). Los que me conocen saben que soy fiel defensor del software libre, pero no es éste el sitio para hablar de las ventajas de usar este tipo de software... únicamente quería apuntar algunas ventajas de GPG sobre PGP.

La versión 1.0.0 de GPG fue liberada el 7 de Septiembre de 1999, y desde entonces han continuado publicándose versiones actualizadas. Actualmente las últimas versiones disponibles son la 1.4.0 de la rama estable y la 1.9.15 de la rama de desarrollo. Como en otros casos de software crítico, recomiendo encarecidamente optar siempre por la versión estable.

A la hora de realizar las prácticas con GPG, podréis seguirlas con cualquier sistema soportado (osea, que los usuarios de Windows podéis también disfrutar de GPG :-P), si bien la parte en que tratamos las diversas interfaces gráficas está dedicada para los usuarios de sistemas Unix-like (y más concretamente, GNU/Linux). Los "windowseros" que no se me quejen, que el de PGP fue entero para ellos... ;-)

Las versiones que yo he utilizado para desarrollar el artículo son:

- GnuPG 1.4.0-2 bajo GNU/Linux Debian SID.
- GPA 0.7.0-1 bajo GNU/Linux Debian SID.
- Kgpg 3.3.2-1 bajo GNU/Linux Debian SID.
- GnuPG 1.4.0 bajo Microsoft Windows XP SP1. (¿Pero alguien usa el Spyware Pack 2? :-D)

Para instalar GPG hay dos opciones: bajar los binarios precompilados, o bien compilar nosotros mismos el código fuente de la aplicación.

Lo más cómodo es utilizar los binarios precompilados (los que seáis debianitas como yo... apt-get install gnupg :-P), pero aún así vamos a ver cómo compilar nosotros mismos el programa en sistemas Linux.

Compilando GnuPG

Lo primero es bajar el paquete con las fuentes de la siguiente dirección:

ftp://ftp.gnupg.org/gcrypt/gnupg/gnupg-1.4.0.tar.gz

> *Para comprender mejor los comandos de consola que vamos a utilizar, es imprescindible que conozcáis al menos por encima el significado del prompt de la shell de Linux. En mi artículo os encontraréis con dos tipos de ellos:*
>
> ***master@blingdenstone:~$***
>
> *Podemos dividir este prompt en tres partes:*
>
> *La primera son los datos del usuario: "master", que es el usuario que está ejecutando la shell; "blingdenstone" que es el dominio en el que se encuentra el usuario (en este caso, el nombre de la máquina); y el símbolo "@" que denota pertenencia. Así pues "master@blingdenstone" significa "esta sesión es del usuario master, perteneciente al dominio blingdenstone".*
>
> *El símbolo ":" denota la separación entre las dos partes del prompt.*
>
> *La segunda parte del prompt son los datos relativos a la sesión: en primer lugar tenemos el directorio de trabajo y que en este caso es "~", símbolo con el que denotamos al directorio home del usuario que ejecuta la shell; y luego un símbolo que puede ser "$" ó "#" para usuarios sin privilegios y con privilegios respectivamente.*
>
> *La tercera parte es el propio comando que nosotros introducimos, y que puede ser lo que nos dé la gana.*
>
> *Por tanto, este prompt significaría a grandes rasgos "el usuario master, perteneciente al dominio blingdenstone, que se encuentra en su directorio home y que no dispone de privilegios especiales, desea ejecutar...".*
>
> *El otro tipo de prompt que veréis es:*
>
> ***blingdenstone:/home/master#***
>
> *Es igual que el anterior pero con un pequeño detalle: no especifica el usuario que está ejecutando la shell. Esto es así porque (y ésto nos lo indica el símbolo "#") el usuario en este caso es el root de la máquina, y por tanto se denota su prompt únicamente con el nombre de la misma.*
>
> *Este prompt significaría "el root del dominio blingdenstone, que se encuentra en /home/master, desea ejecutar...". No indicamos que se trata de un usuario privilegiado porque se sobreentiende que el root siempre lo es (para los más frikis: sí, sé que puede no ser siempre así... :-P).*
>
> *Estos dos tipos de prompt son los que encontraréis en una distribución Debian (la que yo uso) o cualquiera de sus derivadas. Para otras distribuciones (como SuSe, Fedora, Mandrake...) puede variar ligeramente.*

Ahora que ya lo tendréis en el disco duro, vamos a aprovechar para practicar cosas ya vistas. Comprobemos que tenemos todos el mismo paquete para compilar... para ello aquí os dejo el hash MD5 y SHA-1 del paquete. Que cada cual elija el que más le guste (podéis usar md5sum y sha1sum para comprobar los respectivos hashes en Linux):

```
master@blingdenstone:~$ ls -l gnupg-1.4.0.tar.gz
-rw-r--r--  1 master master 3929941 Mar  1 00:03 gnupg-1.4.0.tar.gz
master@blingdenstone:~$ md5sum gnupg-1.4.0.tar.gz
74e407a8dcb09866555f79ae797555da  gnupg-1.4.0.tar.gz
master@blingdenstone:~$ sha1sum gnupg-1.4.0.tar.gz
7078b8f14f21d04c7bc9d988a6a2f08d703fbc83  gnupg-1.4.0.tar.gz
master@blingdenstone:~$
```

Bien, ahora procederemos a descomprimirlo con el siguiente comando:

```
master@blingdenstone:~$ tar xvfz gnupg-1.4.0.tar.gz
```

Ahora debemos situarnos en el directorio que contiene las fuentes:

```
master@blingdenstone:~$ cd gnupg-1.4.0
master@blingdenstone:~/gnupg-1.4.0$
```

Observad cómo ha cambiado el directorio de trabajo...

Bien, llegados a este punto siempre es bueno hacer un ls y echar un vistazo a los ficheros que nos encontramos para buscar el que contenga las instrucciones de instalación del programa (que normalmente se llamará INSTALL), pero en nuestro caso os voy a ahorrar el trabajo: hay que ejecutar el script de autoconfiguración:

```
master@blingdenstone:~/gnupg-1.4.0$ ./configure
```

Y aquí comienza una larga lista de comprobaciones automáticas que sirven para determinar si tu sistema está preparado para hacer funcionar el programa que pretendes compilar, así como la configuración de compilación necesaria para el mismo. Este paso es necesario para evitar que compiles a lo loco el programa y luego te encuentres con que necesitas una librería que no está instalada.

Los errores en esta parte pueden ser de lo más variopintos, y no tienen nada que ver con el artículo, por lo que si tienes cualquier problema con este punto, te recomiendo que te pases por nuestro foro, y allí preguntes indicando detalladamente el error obtenido (copiar y pegar la salida por consola es lo más fácil). Seguro que encuentras mucha gente dispuesta a echarte una mano.

> *Si aún no conoces nuestro foro, no dejes de visitarlo:*
> *http://www.hackxcrack.com/phpBB2/*

Si todo ha ido bien, deberías ver al final unas líneas parecidas a estas:

```
config.status: creating po/Makefile
config.status: executing g10defs.h commands
g10defs.h created

        Configured for: GNU/Linux (i686-pc-linux-gnu)
```

Eso significa que estamos listos para compilar, cosa que vamos a hacer ahora mismo:

```
master@blingdenstone:~/gnupg-1.4.0$ make
```

Y si lo del ./configure os parecieron muchas líneas, esperad a ver esto... :-D

El comando make compila todos los ficheros de código del programa según las reglas definidas en el fichero Makefile (que a su vez ha sido generado por el script de configuración anterior).

Si todo ha ido bien, veremos algo como así:

```
make[2]: Leaving directory `/home/master/gnupg-1.4.0/checks'
make[2]: Entering directory `/home/master/gnupg-1.4.0'
make[2]: Nothing to be done for `all-am'.
make[2]: Leaving directory `/home/master/gnupg-1.4.0'
make[1]: Leaving directory `/home/master/gnupg-1.4.0'
master@blingdenstone:~/gnupg-1.4.0$
```

Ahora debemos "instalar" el programa que acabamos de compilar, para lo cual primero debemos "hacernos root" mediante el comando su y después ejecutar la orden de instalación en el directorio de binarios del sistema (/usr/bin). El que tengamos que hacernos root es debido a que dicho directorio requiere privilegios especiales para escribir en él.

```
master@blingdenstone:~/gnupg-1.4.0$ su
Password:
blingdenstone:/home/master/gnupg-1.4.0# make install
```

Y unas cuantas líneas más. :-P

Ahora lo primero que debemos hacer es volver a convertirnos en nuestro usuario habitual (mediante el comando exit) y comprobar (desde cualquier directorio) que GPG está correctamente instalado:

```
master@blingdenstone:~$ gpg --version
gpg (GnuPG) 1.4.0
Copyright (C) 2004 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.

Home: ~/.gnupg
Supported algorithms:
Pubkey: RSA, RSA-E, RSA-S, ELG-E, DSA
Cipher: 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH
Hash: MD5, SHA1, RIPEMD160, SHA256, SHA384, SHA512
Compression: Uncompressed, ZIP, ZLIB
master@blingdenstone:~$
```

¡Perfecto! Ya tenemos GPG compilado para nuestro sistema y listo para funcionar.

> *Para las "mentes inquietas", que sé que hay muchas por ahí sueltas... echad un ojo a los ficheros contenidos en la carpeta gnupg-1.4.0/cipher/ ... os garantizo que no os aburriréis (si sabéis C, claro).*

Comenzando a trabajar con GPG

Supongo que habréis notado que el nivel en general de este segundo artículo es algo más alto que el del primero. No es casualidad: deseo, poco a poco, ir detallando tanto la teoría como las prácticas, para que la curva de aprendizaje sea suave pero ascendente. Ahora, momento de empezar a conocer GPG, notaréis especialmente esto que acabo de comentar... espero que nadie se me pierda.

Como los conceptos teóricos ya están asimilados del artículo anterior con PGP, no voy a volver a explicar en qué consiste cada uno de los procesos que se llevan a cabo tras las acciones del software. Tampoco voy a detallar de forma exhaustiva todas y cada una de las opciones de GPG, pues eso requeriría mucho más espacio del que disponemos. Vamos, pues, a aprender a usar GPG mientras profundizamos en los conceptos teóricos de la criptografía...

La primera vez que ejecutéis GPG os generará el directorio que contendrá el anillo de claves y la configuración del software. Por defecto, ese directorio será .gnupg dentro del home del usuario.

> *Recordemos que el anillo de claves es el conjunto de todas las claves públicas y privadas que tenemos a disposición del software. El concepto de anillo de claves toma especial importancia en procesos como la firma digital de claves públicas y el establecimiento de niveles de confianza.*

Vamos a comenzar por crearnos nuestra propia clave con GPG...

```
master@blingdenstone:~$ gpg --gen-key
gpg (GnuPG) 1.4.0; Copyright (C) 2004 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.

Please select what kind of key you want:
   (1) DSA and Elgamal (default)
   (2) DSA (sign only)
   (5) RSA (sign only)
Your selection?
```

Entre las opciones disponibles, voy a escoger RSA para la clave nueva que vamos a generar. Los motivos son dos: el primero que el sistema RSA siempre ha sido mi preferido, y el segundo lo averiguaréis más tarde...

```
Your selection? 5
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
```

¿Qué clase de criptomaníacos seríamos si eligiéramos menos de 4096? :-P

```
What keysize do you want? (2048) 4096
Requested keysize is 4096 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)
```

Yo voy a elegir hacer una clave que no expire, pero creo que podréis ver perfectamente que el proceso para generar una clave con fecha de caducidad es trivial.

```
Key is valid for? (0) 0
Key does not expire at all
Is this correct? (y/N)
```

Confirmamos la orden.

```
Is this correct? (y/N) y

You need a user ID to identify your key; the software constructs the user ID
from the Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name:
```

Ahora es el momento de introducir el identificador que deseamos para nuestra clave...

```
You need a user ID to identify your key; the software constructs the user ID
from the Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: Wadalberto HxC
Email address: wadalberto@hackxcrack.com
Comment: -<|:-P
You selected this USER-ID:
    "Wadalberto HxC (-<|:-P) <wadalberto@hackxcrack.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
You need a Passphrase to protect your secret key.

Enter passphrase:
```

Y ahora deberemos introducir (por duplicado en estos casos, como de costumbre) el passphrase que deseamos asignar a la clave. Tras ello, el software comenzará la generación de la clave, para lo cual pide al usuario que genere entropía en la máquina.

> *Como ya comenté en el primer artículo, un computador NO es capaz de generar datos aleatorios, sino únicamente pseudoaleatorios. La única forma de introducir en procesos críticos (como la generación de una clave de cifrado) entropía es que ésta sea introducida desde fuera del sistema por el propio usuario.*

```
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
.+++++
.....................+++++
gpg: key D2AB36BB marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: public key E632B133 is 5708 seconds newer than the signature
gpg: 3 marginal(s) needed, 1 complete(s) needed, classic trust model
gpg: depth: 0  valid:  20  signed:  27  trust: 0-, 0q, 0n, 0m, 0f, 20u
gpg: depth: 1  valid:  27  signed:   0  trust: 0-, 0q, 0n, 14m, 13f, 0u
pub   4096R/D2AB36BB 2005-03-09
      Key fingerprint = 0EF3 77B9 ADDF F172 4695  92F0 EA34 2D4E D2AB 36BB
uid                  Wadalberto HxC (-<|:-P) <wadalberto@hackxcrack.com>
```

```
Note that this key cannot be used for encryption.  You may want to use
the command "--edit-key" to generate a secondary key for this purpose.
master@blingdenstone:~$
```

En la información volcada a pantalla tras la generación de la clave podemos observar algunos datos de gran interés, como la KeyID (D2AB36BB), el tamaño de clave, el fingerprint... todos estos elementos deberían ser de sobra conocidos pues se explicaron en detalle en el artículo anterior.

Pero, como nos muestra el propio GPG, nuestra clave no está completa porque aún no tenemos una subclave de cifrado, por lo que deberemos crearla:

```
master@blingdenstone:~$ gpg --edit-key d2ab36bb
gpg (GnuPG) 1.4.0; Copyright (C) 2004 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.

Secret key is available.

pub  4096R/D2AB36BB  created: 2005-03-09  expires: never      usage: CS
                     trust: ultimate      validity: ultimate
[ultimate] (1). Wadalberto HxC (-<|:-P) <wadalberto@hackxcrack.com>

Command>
```

Y seleccionamos la opción de añadir una nueva subclave.

```
Command> addkey
Key is protected.

You need a passphrase to unlock the secret key for
user: "Wadalberto HxC (-<|:-P) <wadalberto@hackxcrack.com>"
4096-bit RSA key, ID D2AB36BB, created 2005-03-09

Please select what kind of key you want:
   (2) DSA (sign only)
   (4) Elgamal (encrypt only)
   (5) RSA (sign only)
   (6) RSA (encrypt only)
Your selection?
```

Dado que deseamos crear una subclave de cifrado para RSA, seleccionamos la opción 6 y seguimos el mismo proceso para generar la subclave que seguimos para generar la clave principal.

```
Please select what kind of key you want:
   (2) DSA (sign only)
   (4) Elgamal (encrypt only)
   (5) RSA (sign only)
   (6) RSA (encrypt only)
Your selection? 6
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048) 4096
Requested keysize is 4096 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) 0
Key does not expire at all
Is this correct? (y/N) y
Really create? (y/N) y
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
.........+++++
......+++++

pub  4096R/D2AB36BB  created: 2005-03-09  expires: never       usage: CS
                trust: ultimate      validity: ultimate
sub  4096R/4F05B850  created: 2005-03-09  expires: never       usage: E
[ultimate] (1). Wadalberto HxC (-<|:-P) <wadalberto@hackxcrack.com>

Command> quit
Save changes? (y/N) y
master@blingdenstone:~$
```

Ya tenemos creada nuestra clave RSA. Podemos verla dentro de nuestro anillo de claves:

```
master@blingdenstone:~$ gpg --list-keys
/home/master/.gnupg/pubring.gpg
------------------------------
{...}

pub   4096R/D2AB36BB 2005-03-09
uid                  Wadalberto HxC (-<|:-P) <wadalberto@hackxcrack.com>
sub   4096R/4F05B850 2005-03-09

master@blingdenstone:~$
```

He cortado la salida por pantalla porque a nadie le interesarán las otras 45 claves de mi anillo... :-P

Bien, como ya hemos dicho, la pareja de claves RSA ya ha sido creada, pero... ¿qué es RSA en realidad?

Metiendo mano a RSA

Creo que ha llegado el momento de saber realmente cómo funciona un algoritmo de cifrado. El motivo de haber seleccionado RSA y no DSA para la generación de nuestro par de claves es que RSA se calcula con operaciones matemáticas sencillas (sumas, restas, multiplicaciones, divisiones, módulos...) mientras que en DSA intervienen logaritmos, lo cual complica mucho más el cálculo.

Ahora veamos cuáles son los pasos para generar una clave de cifrado RSA:

1- Escoger dos números primos muy grandes p y q (secretos) y calcular el número n (público) correspondiente a su producto, n = p * q

2- Escoger la clave de descifrado constituida por un gran número entero d (secreto), que es primo con el número phi(n) (secreto) obtenido mediante: phi(n) = (p-1) * (q-1)

3- Calcular el entero e (público) tal que 1 <= e <= phi(n) , mediante la fórmula: e * d = 1 (mod phi(n))

4- Hacer pública la clave de cifrado (e, n).

Bien, que nadie se asuste que no es tan complicado como parece. Vamos a seguirla paso a paso para construir nuestra propia clave con papel y pluma (algunos aún escribimos con estilográfica).

En primer lugar, elegimos dos números primos... aunque no serán "grandes primos" como dicta el algoritmo, que no queremos morirnos calculando... xD

p = 11
q = 3
=> n = (p*q) = 33

Ahora debemos calcular el número phi(n):

phi(n) = 10 * 2 = 20

Con estos factores calculados, debemos calcular e, teniendo en cuenta que debe cumplir que su máximo común divisor con (p-1) y (q-1) -y, por tanto, con phi(n)- debe ser 1.

mcd (e, p-1) = 1
mcd (e, q-1) = 1
=> mcd (e, phi(n)) = mcd (e, (p-1) (q-1))= 1

e = 3

mcd (3, 10) = 1
mcd (3, 2) = 1
=> mcd (3, 20) = mcd (3, (10) (2))= 1

Ahora debemos seleccionar el número d teniendo especial cuidado de satisfacer las premisas:

d = 7

e * d-1 = (3 * 7) - 1 = 20
20 | phi(n)=20

Ya hemos terminado de calcular las claves y estamos listos para publicarlas:

Clave pública = (n, e) = (33,3)
Clave privada = (n, d) = (33,7)

Bien, la cosa se va poniendo interesante. Es el momento de ver la especificación del algoritmo para cifrar y descifrar:

1- Para cifrar texto, es necesario previamente codificar el texto en un sistema numérico en base b dividiéndolo en bloques de tamaño j-1 de forma que b^(j-1) < n < b^j.

2- Cifrar cada bloque M_i transformándolo en un nuevo bloque de tamaño j C_i de acuerdo con la expresión C_i == M_i^e (mod n).

3- Para descifrar el bloque C_i , se usa la clave privada d según la expresión: M_i == C_i^d (mod n).

Creo que será mucho más fácil de ver con un ejemplo... supongamos que queremos cifrar un mensaje representado por el número 7.

m = 7

El valor cifrado del mensaje 7 según la clave pública (33,3) es el siguiente:

c = m^e mod n = 7^3 mod 33 = 343 mod 33 = 13

Ahora comprobamos que al aplicar la fórmula de descifrado con la clave privada (33,7) obtenemos de nuevo el mensaje original:

m' = c^d mod n = 13^7 mod 33 = 7

Dado que m=m' el mensaje se ha descifrado correctamente y nuestro par de claves funciona a la perfección. :-)

Ahora ya sabéis cómo funciona en realidad vuestra clave de cifrado por dentro, y lo sabéis de forma totalmente exacta. El sistema es exactamente el mismo, solo que en OpenPGP se utilizan números descomunales mientras que yo he elegido si no los más bajos, casi.

Manejando claves en GPG

Aunque dije que no iba a detallar todas las opciones de GPG, sí hay algunas que no quiero dejar de mencionar. Los datos encerrados entre "<>" son opciones variables que debe introducir el usuario.

Importar clave: gpg --import <fichero>

Exportar clave pública: gpg --armor --export <KeyID> <fichero>

Exportar clave privada: gpg --armor --export-secret-keys <KeyID> <fichero>

Exportar a un servidor de claves: gpg --keyserver <Keyserver> --send-key <KeyID>

Recordad que a un servidor de claves únicamente pueden exportarse claves públicas.

Buscar en un servidor de claves: gpg --keyserver <Keyserver> --search-key <KeyID>

Importar desde un servidor de claves: gpg --keyserver <Keyserver> --recv-key <KeyID>

Firmar una clave pública (firma exportable): gpg --sign-key <KeyID>

Firmar una clave pública (firma no exportable): gpg --lsign-key <KeyID>

Cifrar ficheros: gpg --encrypt-files <fichero>

Descifrar ficheros: gpg --decrypt-files <fichero>

Firmar ficheros (MIME): gpg --sign <fichero>

Firmar ficheros (armadura): gpg --clearsign <fichero>

Verificar firmas: gpg --verify <fichero>

Para conocer en profundidad todas las opciones de GPG, lo mejor es que ejecutéis el comando "man gnupg" y os empapéis del manual de GnuPG. Y si tenéis alguna duda, en el foro podéis preguntar tranquilamente.

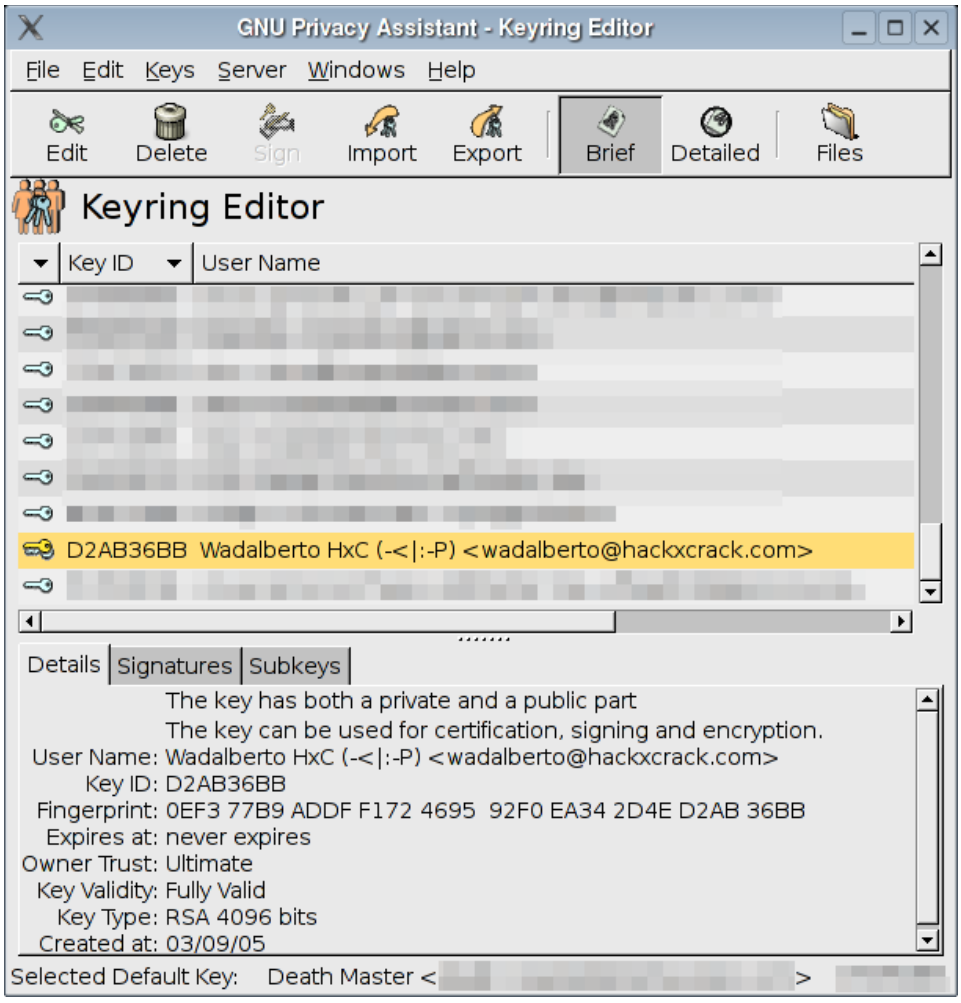
Poniendo cara a GPG

GnuPG es un software muy potente, sin duda, pero la verdad es que trabajar en consola suele resultar bastante engorroso a la mayoría de los usuarios (excepto unos cuantos "tipos raros" como yo), por lo que lo más lógico en los tiempos que corren es que todo programa cuyo uso requiera comandos de consola, tenga una o varias interfaces gráficas de usuario (GUI, del inglés Graphic User Interface) que faciliten al usuario su manejo.

Para GnuPG existen multitud de GUIs para GnuPG, pero personalmente os recomiendo dos de ellas: Gnu Privacy Assistant (GPA) y KGpg. Podéis ver la lista oficial de GUIs de GnuPG en:

http://www.gnupg.org/(es)/related_software/frontends.html#gui

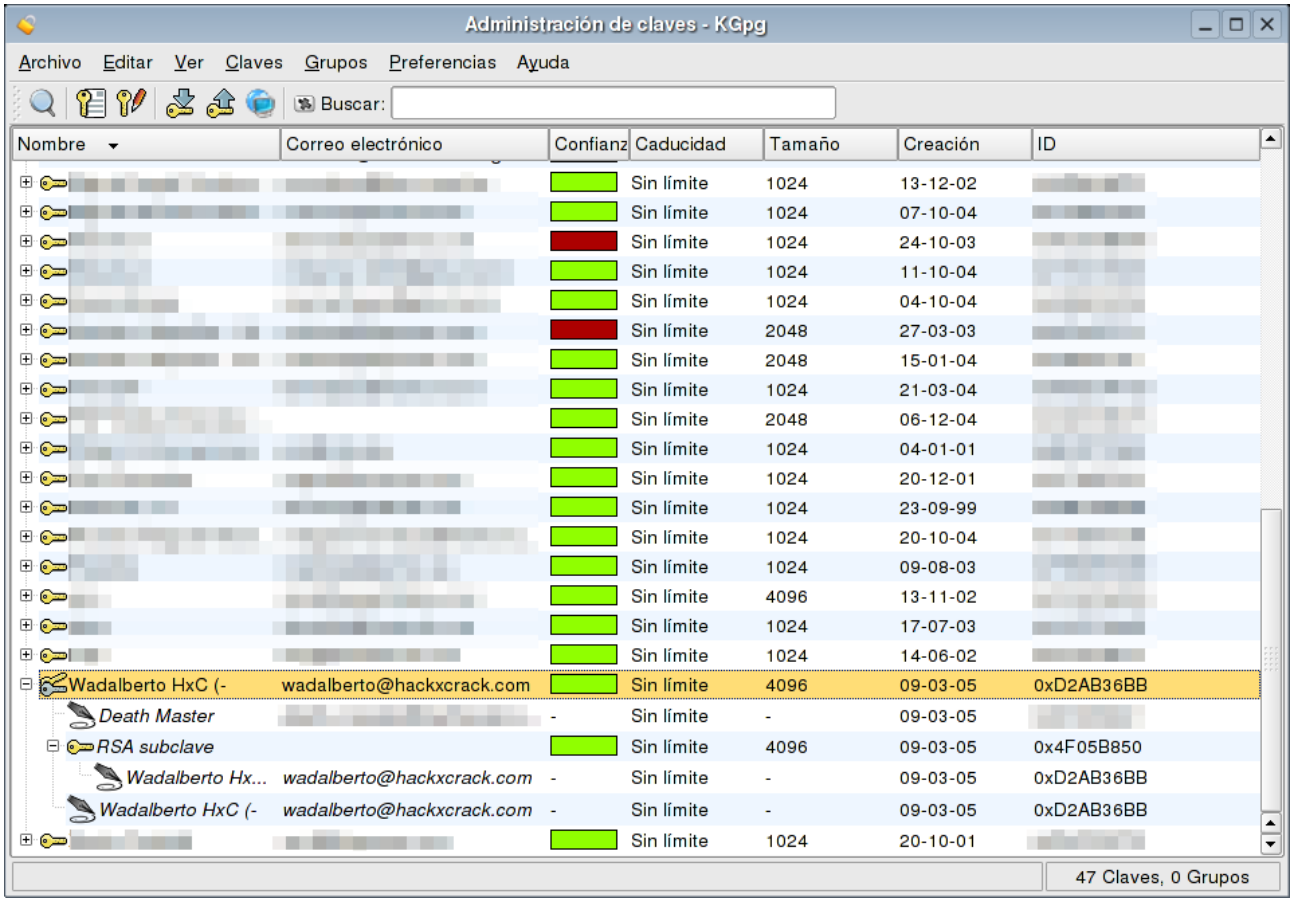
GPA es la primera GUI que usé para GnuPG. Es bastante completa y rápida, si bien no contempla todas las opciones de GPG (cosa que, por otro lado, es común en todas las GUIs de programas de línea de comandos). Está formado por dos partes: el editor del anillo y el gestor de ficheros.



KGpg es la GUI de KDE para GPG. Es la GUI que uso ahora mismo y me parece la más completa que hay hoy por hoy. Su estética es muy parecida a la de PGP para Windows, lo cual es más útil para usuarios que pretendan migrar a Linux, además de la calidad de la misma. Posee muchos más y mejores diálogos y opciones que GPA, además de ser más intuitivo en el uso.

Pero la gran ventaja de KGpg es la integración con KDE (para los usuarios de KDE como yo, claro). En primer lugar, KGpg se integra con Konqueror para permitir trabajar con cifrado de archivos desde el navegador de ficheros. Además, KGpg genera un icono en la bandeja de sistema que permite trabajar con el contenido del portapapeles y GPG de forma transparente al usuario. Por último, KGpg integra la "destructora de documentos" en el escritorio, un sistema de borrado seguro de datos para GnuPG. En definitiva, que KGpg ofrece una experiencia de integración en Linux similar a PGP en Windows, lo cual resulta mucho más agradable para cualquier usuario, además de evitar ese efecto "campo de fuerza" que efectúan algunos programas de línea de comandos en muchas personas... :-P

Para instalar cualquiera de ellos dos, debéis seguir el mismo sistema que hemos seguido con GnuPG, pudiendo elegir paquetes precompilados o compilarlo vosotros mismos (los pasos son prácticamente los mismos que hemos visto al principio del artículo).



Como ejercicio os queda practicar todo lo que sabéis de OpenPGP, y más concretamente de GnuPG, en la interfaz gráfica que prefiráis. A estas alturas deberíais de ser capaces de saber perfectamente para qué sirve cada opción, y haciendo paralelismos con lo visto en PGP, no será complicado que os hagáis con el manejo rápidamente. En cualquier caso, en caso de duda, pregunta en el foro. :-)

Borrado seguro de datos

Como prometí equiparar a los "windowseros" y a los "linuxeros", os voy a recomendar un software de borrado seguro de datos para Linux que, aunque no tiene nada que ver con GnuPG, os servirá para obtener la misma funcionalidad que ofrece el programa wipe de PGP. Ya sé que KGpg ofrece un servicio parecido también, pero habrá gente que no le guste KGpg y prefiera GPA o cualquier otro...

El software se llama wipe (sí, también, ¡pero no es lo mismo!) y se maneja desde consola. La página del programa es:

http://wipe.sourceforge.net/

Su uso es muy sencillo, basta con invocar el comando wipe seguido del fichero a eliminar (con su ruta correspondiente, claro) y confirmar la orden. Claro que siempre hay unas consideraciones que no debemos olvidar: en primer lugar, que para ejecutar este programa es necesario ser el usuario root (ya hemos dicho al principio del artículo como hacernos root); y en segundo lugar, lo mismo que dije para el wipe de PGP: cuidado con lo que eliminéis porque lo que borréis con este método no volverá jamás. Avisados estáis. :-P

Ahora vamos a ver un ejemplo práctico del uso de wipe:

```
master@blingdenstone:~$ su
Password:
blingdenstone:/home/master# wipe heap.c
Okay to WIPE 1 regular file ? (Yes/No) yes
Operation finished.
1 file wiped and 0 special files ignored in 0 directories, 0 symlinks removed but not followed, 0 errors occured.
blingdenstone:/home/master# exit
exit
master@blingdenstone:~$
```

Sencillísimo, ¿verdad?

Terminando...

Este es el fin del segundo artículo del Taller de Criptografía. Espero que os haya gustado (y si es posible, más que el anterior, jejeje).

Sé que lo he dicho muchas veces, pero todas las que se diga son pocas: si tenéis cualquier problema, visitad nuestro foro, donde hay mucha gente dispuesta a ayudar. Y si tenéis alguna sugerencia, podéis escribirme un correo electrónico (las dudas mejor en el foro, donde todos puedan beneficiarse de las respuestas). Algunos pensarán "este tío nos dice que le mandemos un correo y no nos ha dado la dirección...".

En realidad a estas alturas el que no sea capaz de encontrar mi clave pública en algún servidor de claves, es que no se ha leído los artículos y por tanto no puede tener dudas sobre los mismos. ;-P

El próximo número nos meteremos de lleno en cómo aplicar OpenPGP (en cualquiera de sus implementaciones) a las tareas cotidianas frente a la pantalla: copias de seguridad de ficheros, correo electrónico, mensajería instantánea...

Sed buenos.

3- Criptografía y correo electrónico

(Publicado en "PC Paso a Paso: Los cuadernos de Hack X Crack" número 29, 13 de Junio de 2005)

Bienvenidos una vez más al Taller de Criptografía. En los dos primeros artículos hemos aprendido los fundamentos del sistema OpenPGP, los fundamentos matemáticos tras la criptografía de clave pública, así como el manejo de sus principales implementaciones. Ahora es el momento de empezar a darle un uso útil a todos esos conocimientos... ;-)

Antes de empezar, me gustaría decir algo. Cuando empecé esta serie de artículos, decidí que sería un "taller" y no un "curso" porque la mayor parte de trabajo recaería en práctica, dejando la teoría como herramienta para comprender la primera. Dado que el tema a tratar era la criptografía, no me calenté mucho la cabeza y elegí el nombre más obvio para el taller: "Taller de Criptografía".

Ya habiendo terminado el segundo artículo caí en la cuenta de una cosa: sin darme cuenta había "copiado" un nombre que ya existía y que significa mucho para todos los aficionados a la criptografía: El "Taller de Criptografía" de Arturo Quirantes Sierra (profesor de Física de la Universidad de Granada... pero como él mismo dice en su web, podemos respirar tranquilos porque no es contagioso :-P): http://www.ugr.es/~aquiran/cripto/cripto.htm.

Todo aficionado a la criptografía que sepa leer castellano y no esté suscrito al boletín ENIGMA (http://www.ugr.es/~aquiran/cripto/enigma.htm) de Arturo Quirantes, es poco menos que un hereje. Yo particularmente soy asiduo lector desde hace mucho tiempo, y os lo recomiendo totalmente.

Aunque no tengo el placer de conocer a Arturo Quirantes en persona, sí que cruzamos unos correos cuando caí en la cuenta de lo desafortunado de la elección del título del taller, y fue más comprensivo aún de lo que yo esperaba. Sólo me expresó su deseo de que la gente no confundiera este taller que tienes en tus manos con el suyo. Pues que quede esta nota como aclaración, al César lo que es del César. Mando un saludo para Arturo desde mi humilde rincón. :-)

Seguridad y correo electrónico

Si sois aficionados a leer textos sobre seguridad informática, seguramente más de una vez hayáis leído aquello de que el correo electrónico es como una postal y que cualquiera puede leerlo por el camino. Y es cierto. De hecho, no solamente es cierto para correo electrónico (asumo que hablamos de protocolos estándar como SMTP, POP o IMAP, no de webmails y demás engendros de la naturaleza :-P) sino que lo es para casi cualquier protocolo de la red, incluyendo HTTP.

Sin embargo, realizamos muchas operaciones delicadas a través de web y nos han dicho que son seguras... aquello del candadito y demás. Bien, ese "candadito" y el que el habitual HTTP de la barra de direcciones cambie por un HTTPS nos indica que estamos trabajando con SSL (Secure Socket Layer), que es un sistema de cifrado basado en PKI (Public Key Infrastructure)... que, efectivamente, es muy seguro. SSL no es el tema de este artículo... ese tema llegará más adelante, de momento nos basta con saber que existe y que sirve para convertir un protocolo estándar en un protocolo cifrado.

¿Existe algo similar para el correo electrónico? Desde luego. SMTP (puerto 25) se convierte en SMTPS (puerto 465), POP3 (puerto 110) se convierte en POPS (puerto 995), e IMAP (puerto 143) se convierte en IMAPS (puerto 993). Lo malo es que el uso de SSL en correo electrónico está muy poco extendido. Casi ningún servidor de correo gratuito ofrece correo POPS/SMTPS, aunque hay excepciones como Gmail, que aparte de su webmail (¿webmail? ¡arg, engendro! :-D) ofrece la posibilidad de usar sus cuentas en un MUA (Mail User Agent) y los protocolos usados son exclusivamente POPS y SMTP (bajo TLS). Entre los ISP, que yo sepa, casi ninguno ofrece siquiera la posibilidad de usar SSL en el correo, y entre los proveedores privados de alojamiento tampoco son muchos los que ofrecen la posibilidad de usarlo (afortunadamente el mío sí lo hace, yo recibo y envío mi correo a través de SSL).

¿Porqué si tanto nos preocupa usar SSL en HTTP para conexiones delicadas, es por el contrario tan poca la preocupación en el correo electrónico? Pues sinceramente, no lo sé, pero es un hecho. Además, al tratarse SSL de una conexión entre el cliente y el servidor, esta comunicación irá cifrada, sí, pero desde ese servidor al servidor de correo de destino y posteriormente al destinatario, es mucho más que probable que alguna de esas conexiones no se establezca mediante SSL.

Así pues, la mayoría de los usuarios están condenados a mandar sus correos privados (con asuntos tan sensibles como trabajo, datos bancarios, claves de acceso...) como postales a través de la red a la que estén conectados, y que cualquier fisgón de tres al cuarto lea lo que no debe... ¡Pues no!

Vamos a aplicar todo lo aprendido sobre OpenPGP al correo electrónico. Adicionalmente, nunca está de más que uséis, en la medida de lo posible, SSL en detrimento de las conexiones en claro.

En un principio vamos a centrarnos en un MUA concreto para familiarizarnos con el uso de OpenPGP en correo electrónico, pero más adelante veremos otros MUA's alternativos que pueden ser usados de forma muy similar, y finalmente veremos que podemos trabajar con cualquier MUA que nos apetezca e incluso con interfaces webmail (no, no lo voy a volver a decir... bueno venga, la última ¿eh?... ¡engendrooo! xD).

Eligiendo el arsenal

Como ya he dicho, vamos a empezar por un MUA muy concreto. El software elegido es:

- Mozilla Thunderbird (http://www.mozilla.org/products/thunderbird/). El motivo para elegir este software y no otro es que se trata de software libre, es multiplataforma, y mediante un plugin (del que en unos instantes hablaremos) implementa compatibilidad con GnuPG (que también es software libre y multiplataforma). La última versión disponible es la 1.0.2.

- enigmail (http://enigmail.mozdev.org/). Se trata de un plugin para Mozilla Suite y Mozilla Thunderbird que implementa compatibilidad con inline-PGP (RFC #2440) y PGP/MIME (RFC #3156) mediante GnuPG. La útlima versión disponible es la 0.91.0.

- GnuPG (http://www.gnupg.org/). Espero que después del anterior artículo no tenga que decirle a nadie qué es GnuPG (:-P). La última versión disponible es la 1.4.1 (sí, hay una nueva versión desde el artículo anterior).

Dado que todo el software mencionado es multiplataforma, no hay absolutamente ningún problema en seguir esta práctica tanto desde sistemas GNU/Linux como Microsoft Windows e incluso Mac OS X. Yo particularmente en esta parte usaré un sistema Debian GNU/Linux 3.1 (SID) con Mozilla Thunderbird 1.0.2-2, enigmail 0.91-3 y GnuPG 1.4.0-2, que son las versiones que a día de redactar este artículo hay en los repositorios SID de Debian.

Supongo que ya habréis notado que siempre que puedo, oriento las prácticas al software libre (y si puede ser multiplataforma, mejor que mejor). Además de los motivos técnicos (de menor importancia en este caso, y de los cuales no es el momento de hablar), están los motivos prácticos: no todo el mundo tiene el dinero necesario para comprar las caras (carísimas) licencias de software necesarias para determinados programas,o bien la catadura moral para andar pirateando a troche y moche dicho software. Siempre me ha dado mucha rabia cuando leo algún artículo en el que el autor presupone que todos disponemos del último acojo-editor o acojo-compilador de turno. La vida de estudiante no da para tanto... y hablando de Universidad, otra cosa que me repatea es que los profesores de una Ingeniería (Informática, para más inri) pidan la documentación en "formato Word". Lo siento, tenía que desahogarme... :-P

Bien, vamos al tema. Lo primero que debemos hacer, obviamente, es instalar todo el software necesario. Respecto a la instalación de GnuPG no hay nada que decir, pues en el anterior artículo se trató en detalle. Solamente mencionaré que los usuarios de Linux podéis descargar o bien las fuentes y compilarlas (como fue explicado) o bien algún paquete precompilado, mientras que los usuarios de Windows deberéis bajar el binario precompilado (bueno, las fuentes también pueden ser compiladas, pero os garantizo más de un dolor de cabeza). Los enlaces para descarga son:

Para sistemas Unix-like (fuentes): ftp://ftp.gnupg.org/gcrypt/gnupg/gnupg-1.4.1.tar.bz2
Para sistemas Windows: ftp://ftp.gnupg.org/gcrypt/binary/gnupg-w32cli-1.4.1.exe
Para sistemas Mac OS X: http://prdownloads.sourceforge.net/macgpg/GnuPG1.4.1.dmg?download

Lo siguiente es bajar e instalar Mozilla Thunderbird (también es perfectamente posible usar el gestor de correo de la suite Mozilla). El software posee un instalador muy sencillo (del tipo "siguiente, siguiente, siguiente...") así que no creo que haya problema alguno con ello. He seleccionado la versión 1.0.2 en inglés porque a día de hoy la última versión en castellano es la 1.0, pero podría servir exactamente igual. Los que no se lleven muy bien con la lengua de Shakespeare, ya saben qué hacer (aunque en el artículo haré referencia a los textos de Thunderbird en inglés). Los enlaces para descarga son:

Para sistemas Linux: http://download.mozilla.org/?product=thunderbird-1.0.2&os=linux&lang=en-US
Para sistemas Windows: http://download.mozilla.org/?product=thunderbird-1.0.2&os=win&lang=en-US
Para sistemas Mac OS X: http://download.mozilla.org/?product=thunderbird-1.0.2&os=osx&lang=en-US

La primera vez que iniciéis Thunderbird veréis el asistente que os ayudará a añadir vuestra cuenta de correo. Cualquier cuenta con soporte POP/SMTP (o en su defecto IMAP) sirve, y si no tenéis ninguna, podéis usar una cuenta de Gmail mediante POPS/SMTP.

> *Si no tenéis tampoco cuenta de Gmail, pasaos por este hilo del foro:*
> *http://www.hackxcrack.com/phpBB2/viewtopic.php?t=17929*
> *Y seguro que alguien os mandará una invitación encantados.*
>
> *Aprovecho para recomendar encarecidamente (como hago en cada artículo :-P) que visitéis el foro:*
> *http://www.hackxcrack.com/phpBB2/*

Los pasos para añadir la cuenta mediante el asistente son pocos y muy sencillos, por lo que me limitaré a mencionarlos por encima. Si tenéis algún problema, consultad en google o preguntad en el foro.

- New Account Setup: Seleccionamos cuenta de correo (Email account).

- Identity: Introducimos el nombre con el que deseamos figurar en la cuenta, así como la dirección de correo.

- Server Information: Seleccionamos el tipo de servidor entrante (POP o IMAP) e indicamos el nombre del servidor. Podemos seleccionar si usar una bandeja de entrada global o no (a mí particularmente no me gusta).

- User Names: Debemos indicar el nombre de usuario para correo entrante y correo saliente.

- Account Name: Indicamos el nombre deseado para la cuenta.

- Congratulations!: En este punto hemos finalizado la configuración y podemos revisar los datos introducidos para revisarlos y volver hacia atrás si fuera necesario modificar algo.

También es imprescindible configurar el servidor de correo saliente (en caso de usar IMAP este paso no es necesario), para lo cual iremos al menú Edit > Account Settings e introduciremos los datos en la ficha Outgoing Server (SMTP). No voy a entrar en los detalles de la configuración de la cuenta de correo ni de detalles avanzados como el uso de servidores salientes múltiples e independientes entre cuentas, pues no es el objeto de este texto aprender a usar Thunderbird.

Sí señalaré, no obstante, que es importante señalar las opciones de autenticación y cifrado correctas. En el caso de Gmail, por ejemplo, habría que indicar en el servidor entrante POP la opción "Use secure connection (SSL)", así como TLS en el servidor saliente SMTP.

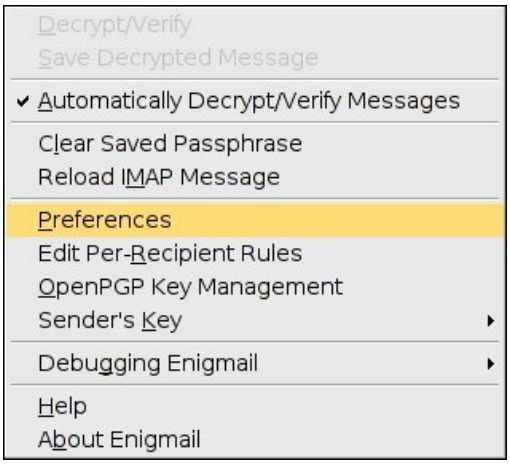
Por último nos queda instalar la "joya de la corona": enigmail. Esta pequeña maravilla programada en XUL (el lenguaje en el que Mozilla vive y respira, del inglés "XML-based User Interface Language") nos va a permitir trabajar con Mozilla Thunderbird y GnuPG implementado de forma transparente al usuario. Lo primero es bajar los ficheros .xpi al disco duro para poder instalarlos desde Mozilla Thunderbird (mucho ojo, si usáis Firefox, con no intentar instalarlos en el mismo... no obstante .xpi es la extensión de las extensiones de Firefox igualmente). Los enlaces para descarga son:

Para sistemas Linux: http://www.mozilla-enigmail.org/downloads/enigmail-0.91.0-tb-linux.xpi
Para sistemas Windows: http://www.mozilla-enigmail.org/downloads/enigmail-0.91.0-tb-win32.xpi
Para sistemas Mac OS X: http://www.mozilla-enigmail.org/downloads/enigmail-0.91.0-tb-darwin.xpi

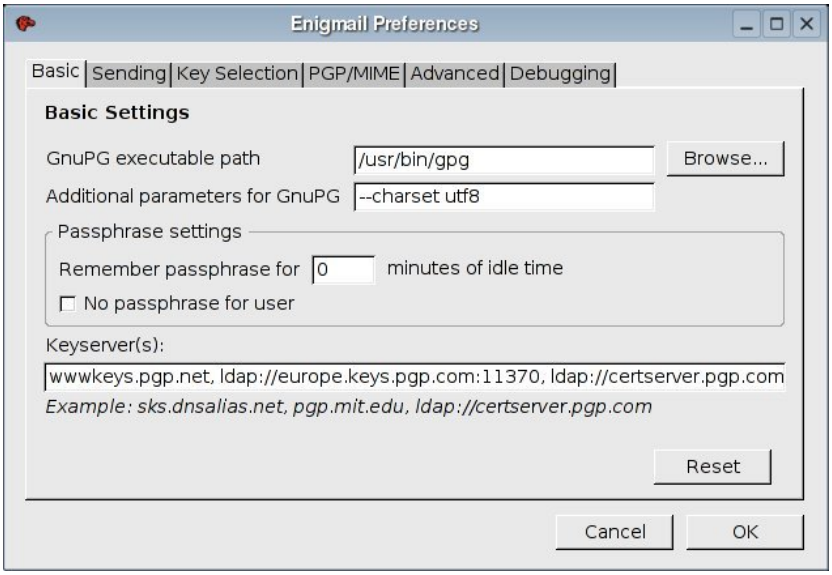
Una vez descargado, debemos instalarlo desde Thunderbird. Para ello iremos al menú Tools > Extensions y seleccionaremos la opción Install, indicando la ruta al fichero .xpi descargado.

Configurando enigmail

Es el momento de meternos en harina con la configuración de enigmail. Para ello, vamos al menú de preferencias (Enigmail > Preferences) y echamos un vistazo a sus distintas pestañas:



- Basic: En este apartado podemos configurar las opciones generales de enigmail. El apartado "GnuPG executable path" nos permite definir la ruta al ejecutable GPG, cosa que puede resultar útil si tenemos instalada más de una versión (yo, por ejemplo, tengo instalada la 1.4.0-3 en /usr/bin/gpg y la 1.9.15-6 de desarrollo en /usr/bin/gpg2) y queremos especificar cuál usar en nuestro correo. En Windows debería ser C:\ruta_a_gpg\gpg.exe. Los parámetros adicionales nos permiten especificar alguna opción que no esté incluida por defecto pero que por algún motivo a nosotros nos interese que sí lo esté (enigmail ya incluye en este apartado el que el set de caracteres sea UTF8). La opción de recordar el passphrase del usuario por un tiempo (en minutos) establecido puede resultar útil cuando se deseen mandar muchos correos en un corto período de tiempo... pero es una opción que yo no recomiendo, por motivos de seguridad. Así mismo, la opción de eliminar la necesidad de passphrase permite trabajar con claves sin passphrase o cuyos usuarios utilicen algún gestor de passphrases como gpg-agent. Por último, en la casilla de servidores de claves podemos especificar cuáles deseamos usar.

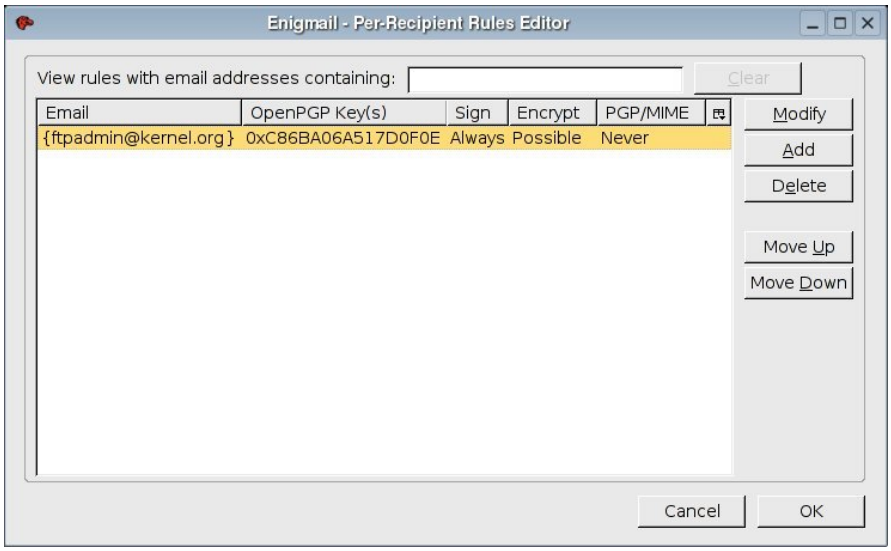


Lamentablemente enigmail ya no soporta PGP, las razones podéis leerlas en el siguiente enlace: http://enigmail.mozdev.org/help.html#win
El equipo de enigmail afirma que la PGP Corp. no ofrece versiones de línea de comandos a un precio razonable, así como tampoco unos sistemas de salida estandarizados.

- Sending: La opción "Encrypt to self" hace que al enviar cualquier correo, nuestra propia clave (la que se encuentre seleccionada en cada cuenta) sea seleccionada como uno de los destinatarios, lo cual es muy útil, pues de otra forma tras enviar un correo y ser éste almacenado en la carpeta de enviados, no podríamos leerlo. La opción "Always trust user ID" hace que se ignore el sistema de confianza de GPG y se permita cifrar a cualquier clave, independientemente de que ésta esté configurada como válida o no. "Always confirm before sending" es autoexplicativa (pide confirmación antes de enviar nada) y es útil cuando tenemos el dedo un poco despistado :-P. La opción "Rewrap signed HTML before sending" reformatea el mail para evitar firmas no válidas en usuarios que utilicen correo HTML (cosa que por otro lado desaconsejo profundamente... con lo bonito que es el texto plano ASCII de toda la vida :-D), pero puede causar que el correo resulte ligeramente modificado. "Allow empty subject" hace que enigmail y Thunderbird no lancen el aviso de asunto vacío que por defecto muestran. Mediante la opción "Allow flowed text" podemos activar el soporte del texto plano fluído que se detalla en el RFC #2646 (ftp://ftp.rfc-editor.org/in-notes/rfc2646.txt).

> *La opción de "Allow flowed text" puede ser útil para usuarios de determinados MUA's que tengan problemas al tratar caracteres como saltos de línea. En Thunderbird no tendréis ningún tipo de problemas de este tipo, por lo que no es necesario que la activéis.*

- Key Selection: En esta pestaña nos encontramos con dos apartados. En el primero de ellos, "Normal Key Selection", podemos seleccionar el comportamiento ante el envío de correo con respecto a las claves: la primera opción es no mostrar nunca la pantalla de selección de claves, de forma que al enviar un correo a un destinatario que no se encuentre en ninguna de las claves que poseemos, enigmail lo enviará sin cifrar sin preguntar; la segunda opción es que la pantalla de selección de claves se muestre cuando sea necesario (el ejemplo dado para la opción anterior); y la tercera es mostrarla siempre. Recomiendo usar la segunda (mostrarla cuando sea necesario), si bien yo la muestro siempre. La segunda parte de esta pestaña configura el comportamiento particular de enigmail según el destinatario del mensaje: desactivar el comportamiento selectivo según destinatario; aplicar las reglas definidas por el usuario en el editor de reglas pre-destinatario, de forma que si no existen reglas no se haga nada; o bien activar las reglas siempre, de forma que si no existe regla alguna predefinida para el destinatario, se cree una nueva. Mi recomendación es activar la segunda opción y, si lo deseamos, definir unas determinadas reglas para un destinatario concreto.



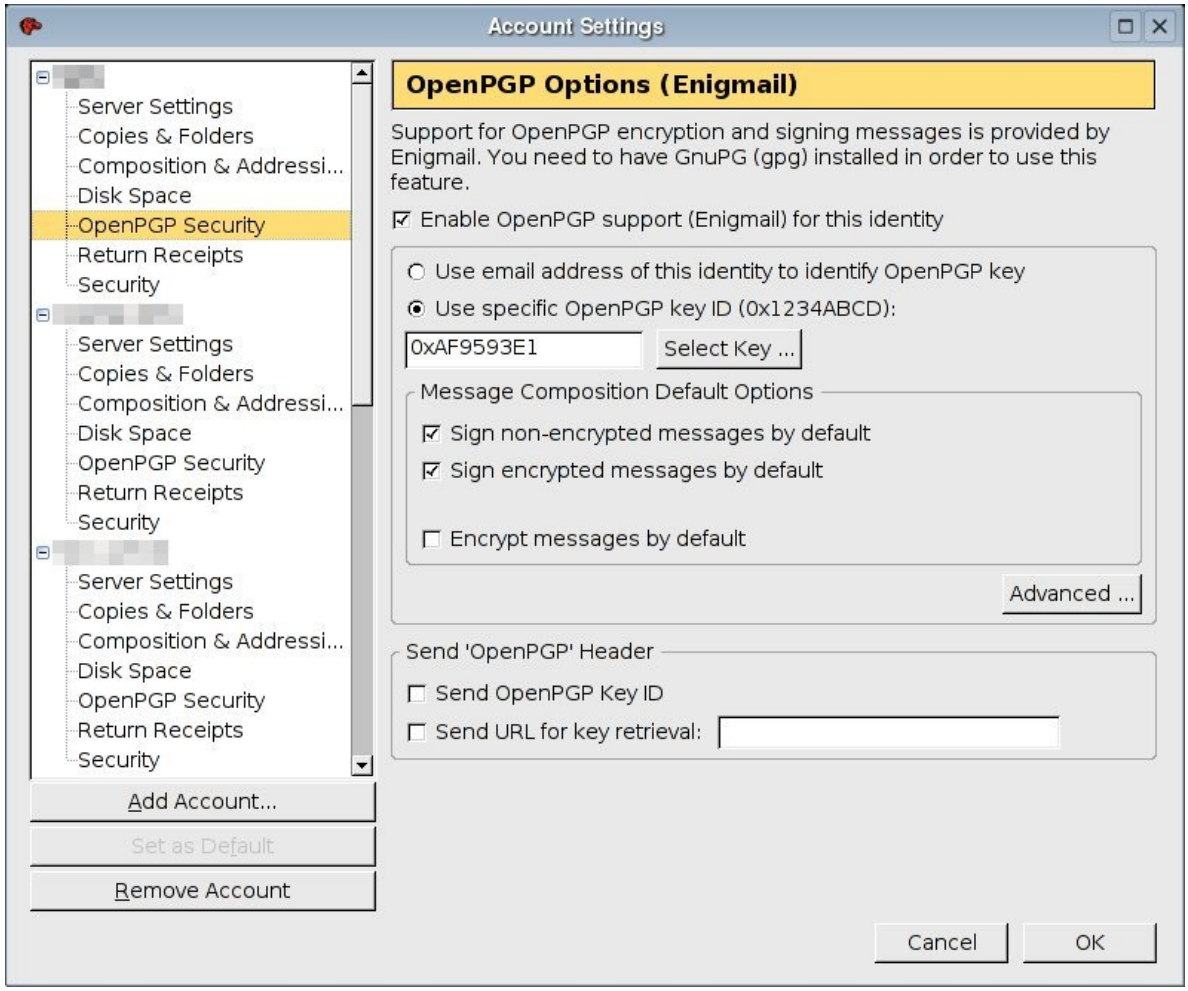
- PGP/MIME: En esta pestaña podemos seleccionar si deseamos usar el estándar PGP/MIME (RFC #3156, ftp://ftp.rfc-editor.org/in-notes/rfc3156.txt), simplemente permitir su uso, o por el contrario no permitirlo. Además, podemos seleccionar el algoritmo de hash que deseamos usar, entre MD5, SHA-1 y RIPEMD-160. Si recordáis el artículo anterior, hasta hace poco SHA-1 era el único que no había sido comprometido, aunque ahora mismo los tres han tenido ataques exitosos a su algoritmia. No obstante, sigo recomendando SHA-1 mientras implementan nuevos y más potentes algoritmos en OpenPGP, además de ser el único que funciona con el 100% de las claves. Respecto al uso de PGP/MIME, lo mejor es permitir su uso y luego usarlo o no según las circunstancias.

- Advanced: En este menú se encuentran las opciones menos comunes de enigmail. "Encrypt if replying to encrypting message" hace que enigmail, por defecto, seleccione la opción de cifrar mensaje cuando estemos respondiendo a un mensaje cifrado, lo cual es altamente conveniente (si nuestro interlocutor considera que la conversación es lo suficientemente importante para estar cifrada, no vamos a contradecirle... :-D). Mediante "Do not add Enigmail comment in OpenPGP signature" evitamos que enigmail incluya en el comentario una frase informando de su uso. Para todos aquellos que acostumbran a adjuntar firmas en su correo electrónico (yo, por ejemplo) o leen correos con firmas, la opción "Treat '--' as signature separator" es importante, pues el estándar openPGP establece que la cadena '--' se transforme en '- -' a la hora de firmar, de forma que si no activamos esta opción no veremos correctamente las firmas (texto gris claro). La opción "Use gpg-agent for passphrase handling" es útil para los que no quieran tener que teclear contraseñas (cosa, por otro lado, poco recomendable y muy insegura) y debe ser usada en conjunción con la opción "No passphrase for user" del menú Basic de configuración. La siguiente opción permite modificar la forma en que enigmail trata las cadenas de direcciones de correo electrónico, eliminando los símbolos '<>' para mantener compatibilidad con claves generadas con Hushmail. La opción "Load MIME parts on demand (IMAP folders)" es recomendable que sea desactivada si usas una o más cuentas IMAP, pues permite a enigmail tratar correctamente los ficheros adjuntos como armadura de texto, evitando el problema que genera Thunderbird al cargar los mensajes según son requeridos por el cliente. Existe una opción adicional (justo antes de la última) si usáis enigmail en Mozilla Suite llamada "Hide SMIME button/menus" que elimina de la interfaz los elementos referidos al estándar S/MIME.

> *S/MIME (Secure/Multipurpose Internet Mail Extensions) es un estándar que propone varias extensiones de seguridad para correo electrónico, basado en el sistema PKI (Public Key Infrastructure) y muy parecido a OpenPGP en su concepción. Actualmente la versión más reciente es la 3, que como siempre, podemos conocer gracias a los correspondientes RFC's:*
> *- S/MIME Version 3 Message Specification (RFC #2633): ftp://ftp.rfc-editor.org/in-notes/rfc2633.txt*
> *- S/MIME Version 3 Certificate Handling (RFC #2632): ftp://ftp.rfc-editor.org/in-notes/rfc2632.txt*
>
> *¿Es mejor o peor que OpenPGP? Va en gustos. Lo que sí es a todas luces es algo más complejo en su estructura (pues aparte de los clásicos algoritmos de cifrado simétricos, asimétricos y de hash, tenemos que tratar con otros elementos como los certificados de seguridad X.509), y algo anticuado en cuanto a los algoritmos usados y siempre comparándolo con OpenPGP (S/MIME utiliza 3DES como algoritmo de cifrado simétrico).*

- Debugging: En este menú se configuran las opciones de corrección de errores de Thunderbird. En el campo "Log directory" podemos configurar una ruta para almacenar los logs de enigmail (en caso de no especificar ninguna ruta, no se almacenarán logs). En el campo "Test email" podremos introducir un email para probar la introducción del passphrase, la validez de la clave...

Ya hemos configurado las opciones generales de enigmail, pero aún nos queda por configurarlo particularmente para la cuenta creada. Para ello iremos a la configuración de la cuenta a través del menú Edit > Account Settings y seleccionamos el menú "OpenPGP Security". Ahora debemos activar el soporte OpenPGP para esta cuenta, y seleccionar el método de elección de clave: en base al correo electrónico o una clave determinada (recomiendo esta segunda opción). Además debemos configurar el comportamiento respecto a firma y cifrado de enigmail con esa cuenta, para lo cual recomiendo seleccionar firmar siempre todos los mensajes (tanto cifrados como no cifrados) pero no activar el cifrado por defecto. Las dos últimas opciones permiten añadir sendos campos a la cabecera del correo electrónico indicando el KeyID de la clave usada y una URL de donde bajarla respectivamente.



La consola de enigmail

Hay un elemento que quiero comentar antes de empezar a trabajar con enigmail: la consola de debugging. Esta consola puede ser accedida mediante el menú Enigmail > Debugging Enigmail > View Console. En ella, al arrancar Thunderbird, solamente se reflejará la información de arranque:

```
Initializing Enigmail service ...
EnigmailAgentPath=/usr/bin/gpg

enigmail> /usr/bin/gpg --charset utf8 --batch --no-tty --version
gpg (GnuPG) 1.4.0
Copyright (C) 2004 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.

Home: ~/.gnupg
Supported algorithms:
Pubkey: RSA, RSA-E, RSA-S, ELG-E, DSA
Cipher: 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH
Hash: MD5, SHA1, RIPEMD160, SHA256, SHA384, SHA512
Compression: Uncompressed, ZIP, ZLIB, BZIP2
```

Ahora veamos qué aparece en la consola cuando envío un correo firmado...

```
enigmail> /usr/bin/gpg --charset utf8 --batch --no-tty --status-fd 2 --comment '
Using GnuPG with Thunderbird - http://enigmail.mozdev.org' --clearsign -u 0xAF95
93E1 --passphrase-fd 0 --no-use-agent
```

Ahora, si introduzco el passphrase de forma incorrecta...

```
enigmail> /usr/bin/gpg --charset utf8 --batch --no-tty --status-fd 2 --comment '
Using GnuPG with Thunderbird - http://enigmail.mozdev.org' --clearsign -u 0xAF95
93E1 --passphrase-fd 0 --no-use-agent
gpg: skipped "0xAF9593E1": bad passphrase
gpg: [stdin]: clearsign failed: bad passphrase
enigmail.js: Enigmail.encryptMessageEnd: Error in command execution
enigmail.js: Enigmail.encryptMessage: Error in command execution
```

Como vemos la consola nos puede ayudar mucho a la hora de controlar los errores producidos, así como para conocer mejor GnuPG (a través del análisis de los comandos que ejecuta enigmail).

Secretos en Thunderbird

Bien, tras todo este rollo teórico y configuraciones varias, ya podemos meternos con la parte práctica... que al fin y al cabo siempre es la más interesante. Vamos a empezar aprendiendo a analizar los correos firmados y/o cifrados que recibimos en nuestro buzón...

Como ejemplo, voy a tomar el correo del boletín una-al-día de la gente de Hispasec (http://www.hispasec.com/, desde aquí mando un saludo para ellos).

Como vemos, nada más seleccionar el correo en la ventana principal de Thunderbird (o bien al abrir el mensaje en una ventana independiente) veremos una sección añadida por enigmail con fondo verde en la que se nos proporciona información sobre el estado de la firma (en caso de no ver la información completa, pulsad sobre el icono "+" junto a la cabecera). Si nada ha fallado, nos mostrará un mensaje similar a este:

```
Good signature from <emisor> <correo_del_emisor>
Key ID: <KeyID_del_emisor> / Signed on: <fecha_de_firma>
```

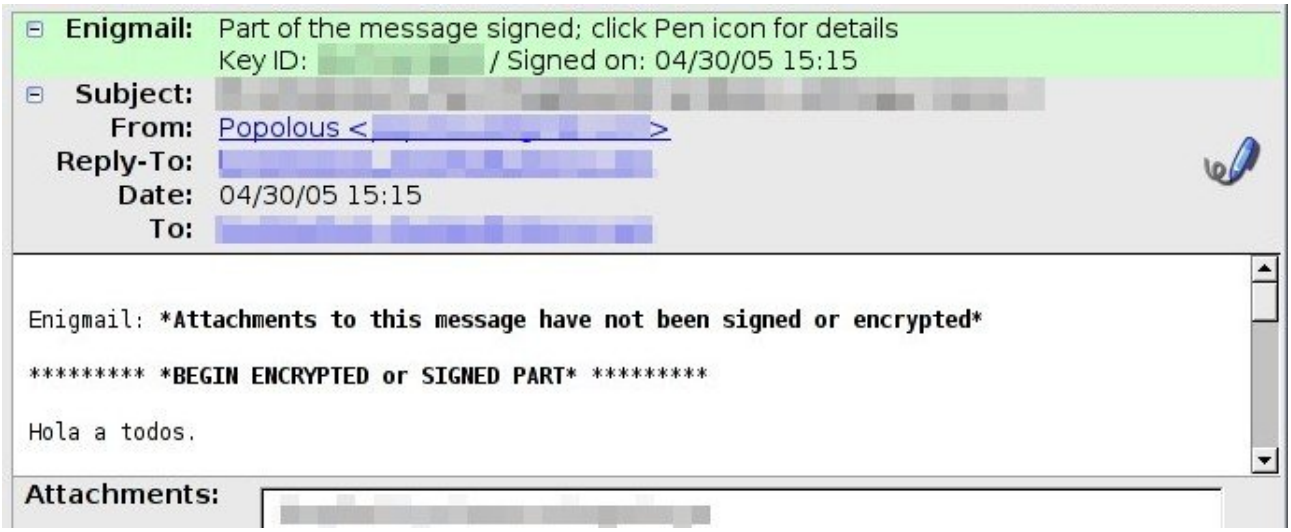
Podemos pulsar en el icono del bolígrafo para obtener información extendida, así como consultar la consola de enigmail para ver qué comando se ha ejecutado y la salida del mismo:


```
enigmail> /usr/bin/gpg --charset utf8 --batch --no-tty --status-fd 2 -d
gpg: Signature made Mon May  2 01:50:47 2005 CEST using RSA key ID FB5671E1
gpg: Good signature from "Noticias Hispasec - 2004 <noticias@hispasec.com>"
```

Todo esto es lo que ocurrirá en la mayoría de las ocasiones, cuando no exista ningún problema con la firma. Pero, como bien sabemos todos, siempre hay algún problema... :-P

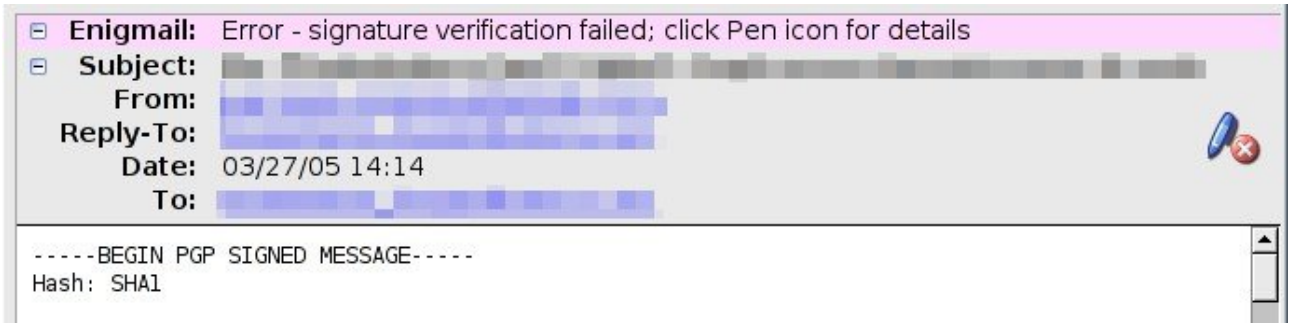
Es bastante común que únicamente una parte del correo haya sido firmado. Esta situación se da, por ejemplo, cuando nuestro interlocutor está respondiendo a un correo firmado (con lo que la parte respondida, y siempre y cuando no haya sido modificada, estará firmada), cuando se adjuntan ficheros que no han sido firmados, o cuando intencionadamente se firma una parte discreta de mensaje.

Veamos un ejemplo. Este correo fue enviado por mi querido amigo Popolous a la lista de correo de gestión de textos de la revista, y aquí vemos un ejemplo de un mensaje parcialmente firmado: enigmail nos advierte de que parte del mensaje ha sido firmado, de que los adjuntos no han sido firmados, y nos especifica con un mensaje llamativo en el cuerpo del correo qué parte es la firmada.



Todas las partes que oculto de las imágenes son para proteger la privacidad de alguien. Mi KeyID ha sido publicada varias veces en anteriores artículos, así como direcciones o KeyID de listas públicas (por ejemplo, una-al-día). Lo que no voy a hacer es publicar ni direcciones ni KeyID privadas de usuarios particulares.

Ahora vamos a ver otro ejemplo en el que la firma está directamente mal (también de Popolous... lo siento, hoy te ha tocado xD). Como vemos, enigmail nos muestra dos avisos visuales llamativos para alertarnos de la situación: en primer lugar el fondo de la cabecera añadida cambia a rojo, con su correspondiente mensaje de error; y además el icono del bolígrafo ahora muestra una señal de error. Pulsando en dicho icono, o bien acudiendo a nuestra amada consola de enigmail (;-P) podremos obtener más detalles del error. Yo consultaré la consola:




```
enigmail> /usr/bin/gpg --charset utf8 --batch --no-tty --status-fd 2 -d
gpg: Signature made Sun Mar 27 14:14:54 2005 CEST using RSA key ID <KeyID>
gpg: BAD signature from "Popolous <correo>"
enigmail.js: Enigmail.decryptMessageEnd: Error in command execution
```

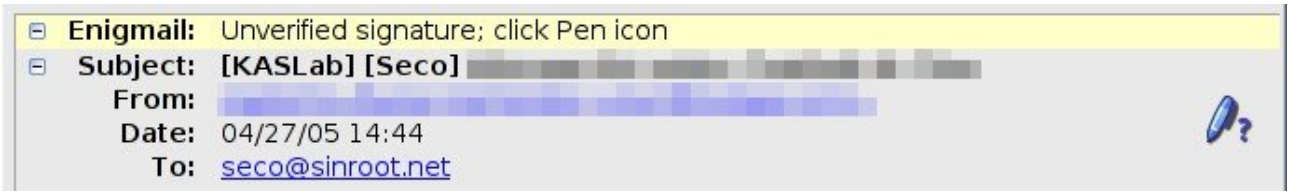
Evidentemente, no siempre una alerta de una firma errónea supone que alguien haya estado enredando en tus correos. Pero cuando todo marcha bien, sí es síntoma inequívoco de que nada malo ha pasado.

Ahora veamos qué pasa cuando recibimos un correo cifrado, por ejemplo uno que me mandó recientemente mi amigo AcidBorg (sí, hay gente que se manda el correo cifrado como costumbre, ¿qué pasa? xD). En primer lugar -evidentemente- nos pedirá el passphrase de nuestra clave como mínimo una vez (pueden ser más, si hay archivos adjuntos que también han sido cifrados, con lo que nos la pedirá tantas veces como archivos haya más una -el correo-). En caso de introducirla mal (o no introducirla), no veremos nada y enigmail nos devolverá como error "bad passphrase" (cuyo ejemplo ya puse antes, hablando de la consola). Una vez descifrado el correo, veremos la cabecera con fondo verde con los mensajes correspondientes a la verificación de la firma y además el del descifrado del correo. Al icono de firma correcta hay que añadir ahora uno de descifrado correcto (representado por una llave), si bien ambos proporcionan la misma información.

La salida de la consola ahora será un poco más larga, pues debe contener la información del proceso de descifrado y de verificación de la firma:

```
enigmail> /usr/bin/gpg --charset utf8 --batch --no-tty --status-fd 2 -d --passph
rase-fd 0 --no-use-agent
gpg: encrypted with 4096-bit RSA key, ID 0260BBB0, created 2003-07-21
     "Death Master <correo>"

enigmail> /usr/bin/gpg --charset utf8 --batch --no-tty --status-fd 2 --verify
gpg: Signature made Mon Apr 25 23:28:02 2005 CEST using DSA key ID <KeyID>
gpg: Good signature from "AcidBorg <correo>"
```

Es posible que en algún momento recibamos un correo firmado por alguien cuya clave no poseemos (si estás suscrito a alguna lista de correo, es muy común). En ese caso, enigmail nos avisará con un mensaje de error con fondo beige en su cabecera y nos instará a pulsar en el icono del bolígrafo para obtener más detalles. He tomado como ejemplo un correo mandado a la lista de correo del Hacklab Vallekas (otro saludo desde aquí para toda la gente del Kaslab, de quienes podéis encontrar más información en http://vallekaslab.ath.cx/) por alguien cuya clave no se encontraba en mi anillo.

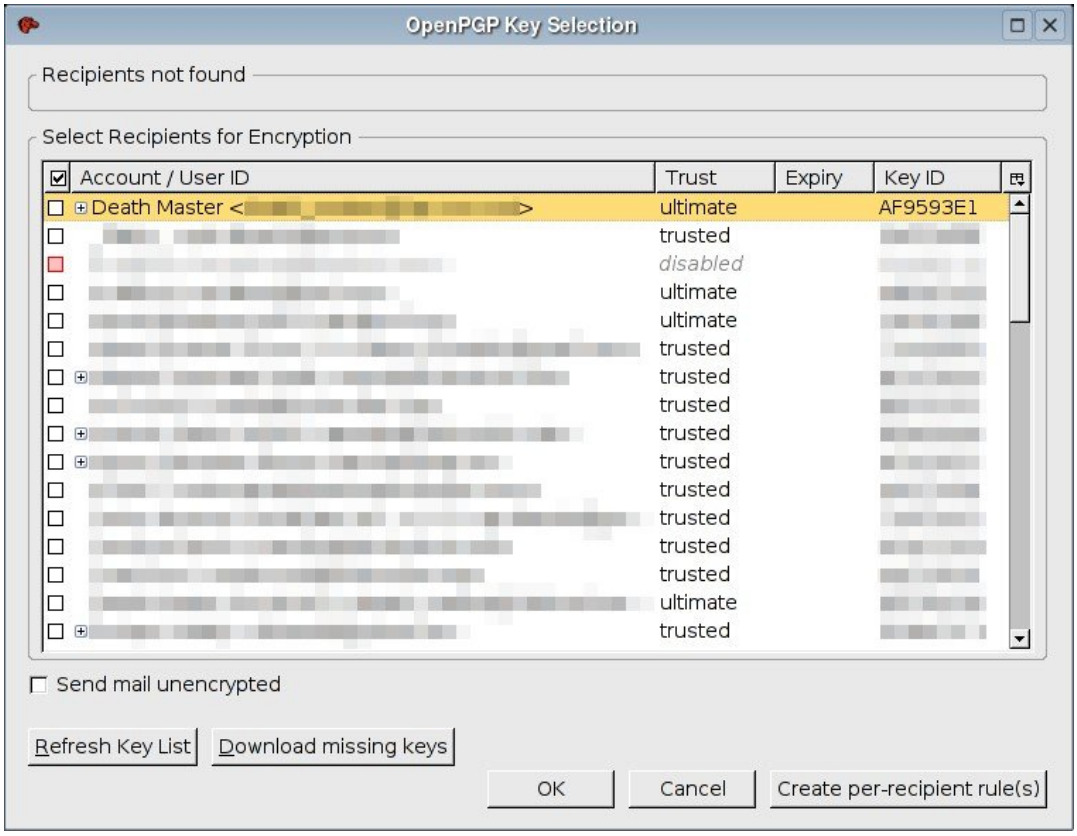


Al pulsar en el icono del bolígrafo enigmail nos avisará de que no ha podido verificar la firma por no encontrar la clave pública del firmante en nuestro anillo, así como nos solicitará permiso para descargar dicha clave de un servidor de claves. Si le otorgamos permiso, nos preguntará qué servidor deseamos usar de entre los que están indicados en las opciones de enigmail, tras lo cual se conectará al mismo y descargará la clave a nuestro anillo directamente.

Enviar correos firmados y/o cifrados es mucho más sencillo, gracias a que enigmail se encarga de la parte dura. ;-)

La firma es totalmente transparente. Simplemente hay que seleccionar en el menú OpenPGP de la barra de tareas la opción "Sign Message" (y si habéis seguido mis recomendaciones sobre la configuración de enigmail, ni siquiera eso). Al dar la orden de envío a Thunderbird, se nos pedirá el passphrase y tras introducirlo correctamente, el correo se firmará y enviará automáticamente.

Para enviar un correo cifrado hay que seleccionar la opción "Encrypt Message" (es recomendable firmar también los mensajes cifrados). Al dar la orden de envío, pueden pasar dos cosas: que el correo se mande automáticamente (o bien os pida el passphrase, en caso de haber seleccionado también la opción de firma), o bien que aparezca la ventana de selección de claves.

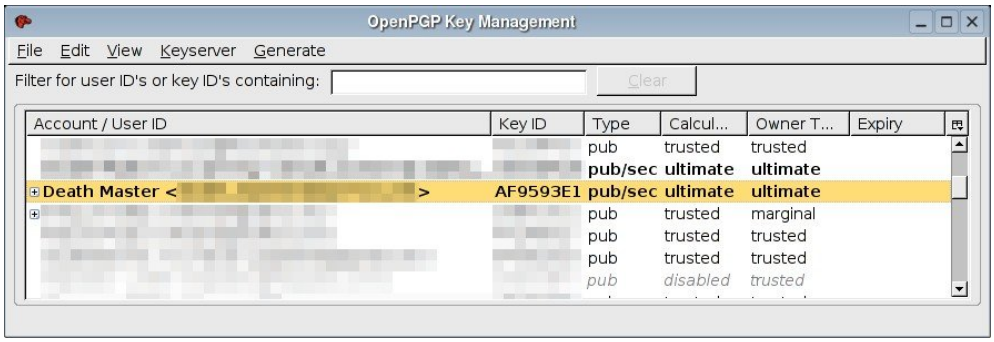


¿En qué condiciones aparece esa ventan? Pues depende de la configuración que hayáis establecido en enigmail. Si habéis seguido mi recomendación, únicamente aparecerá cuando se esté enviando un correo a un destinatario que no figura en ninguna de las claves de nuestro anillo. Si habéis configurado enigmail como yo, aparecerá siempre xD.

En esta ventana simplemente debemos seleccionar el o los destinatarios a los que deseamos cifrar el mensaje y pulsar aceptar. Es posible, así mismo, indicar en este paso que el correo sea mandado sin cifrar (en la casilla "Send mail unencrypted"); refrescar la información del anillo de claves; descargar nuevas versiones de las claves desde un servidor de claves; y editar las reglas de pre-destinatario.

Existen así mismo un par de opciones adicionales en el menú OpenPGP de composición de correo: la primera nos permite activar el uso de PGP/MIME, y la segunda ignorar las reglas pre-destinatario para el mensaje actual.

El administrador de claves de enigmail



Existe en enigmail una opción relativamente reciente (desde la versión 0.89 del 14 de Noviembre de 2004) consistente en toda una interfaz gráfica para GnuPG incrustada en enigmail y el gestor de correo que lo use (sea Thunderbird o Mozilla). Como veréis, tiene un gran parecido con otras GUI's.

Podemos acceder a esta interfaz denominada "OpenPGP Key Manager" mediante el menú Enigmail > OpenPGP Key Management. Esta opción resulta especialmente útil a los usuarios de GnuPG bajo Windows, que de esta forma pueden evitarse el uso de la línea de comandos si les resulta incómoda.

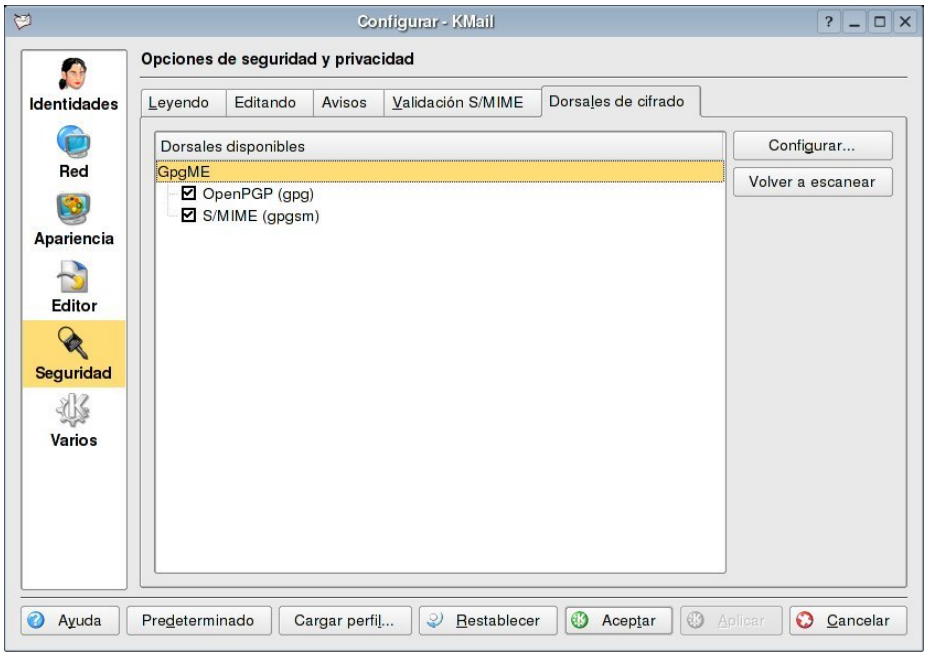
Mediante esta interfaz podemos realizar todas las opciones básicas de GnuPG de una forma similar a cualquier otra GUI: trabajo con claves, acceso al servidor de claves, generación de pares de claves... no dejéis de echarle un vistazo. ;-)

Otros MUA's con soporte OpenPGP

Aunque en mi opinión Mozilla Thunderbird es, hoy por hoy, uno de los mejores clientes de correo que existen, no a todo el mundo tiene porqué gustarle. Además, para los usuarios de Windows supone no poder usar PGP y depender de GnuPG, lo cual no a todo el mundo le gusta (si bien es cierto que con el administrador de claves de enigmail ya no es tan problemático...).

Para todas esas personas, ahora vamos a conocer otros clientes de correo que permiten manejar OpenPGP con gran facilidad. Primero vamos a hablar de los clientes para sistemas Unix-like (como GNU/Linux) y luego para sistemas Microsoft Windows. Cada cual que pruebe y elija el que más le guste. :-)

- Kmail (*NIX): Se trata del cliente de correo integrado en el gestor de ventanas KDE. Implementa OpenPGP con GnuPG mediante GPGME (GnuPG Made Easy). Su configuración y uso es muy similar a todas las que ya hemos visto, por lo que no creo que haya que entrar en detalles.

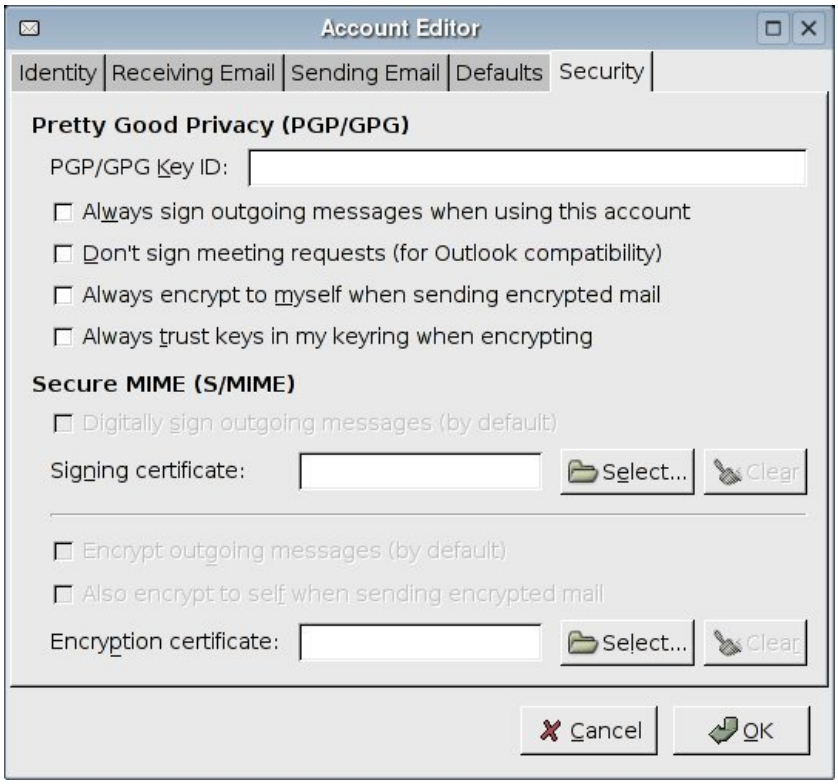


En entornos Unix-like los entornos gráficos tienen dos partes fundamentales: el servidor gráfico y el gestor de ventanas. El servidor gráfico es el "corazón" del sistema gráfico, y existen varias implementaciones (Xfree86, Xorg...), mientras que el gestor de ventanas es el "rostro" del mismo, la parte visible al usuario.

Gestores de ventanas existen muchos: KDE (K Desktop Environment) y GNOME (GNU Network Object Model Environment) son los principales y más utilizados, pero existen muchos otros: Fluxbox, WindowMaker, IceWM, Blackbox, XFCE, Enlightenment...

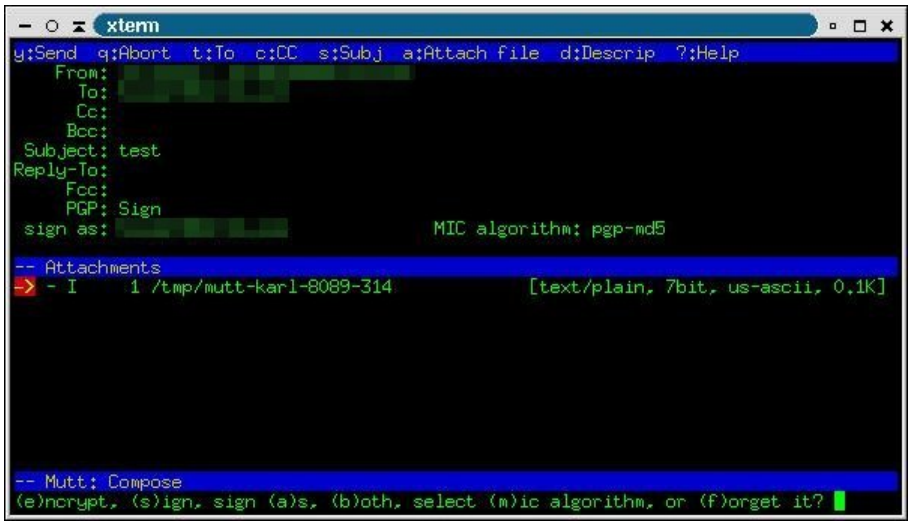
Entre los usuarios de KDE (como yo) y los de Gnome (esos herejes ;-P) siempre ha existido un "sano pique"... aprovecho para mandar desde aquí un saludo para mis amigos Gnomeros, que por cierto son muchos. :-D

- Evolution (*NIX, Windows): Ahora es el turno del cliente integrado en el gestor de ventanas Gnome. La implementación OpenPGP es directamente a través de GnuPG, y aunque no implementa tantas opciones como Kmail o enigmail, son más que suficientes como para manejarse cómodamente con él. Además, existe desde hace relativamente poco una implementación para sistemas Windows, por lo que también los windowseros podréis disfrutar de este cliente (mucha gente dice que es de los mejores... fue uno de los primeros que probé y personalmente no me gustó).



- Sylpheed (*NIX, Windows): Este cliente gráfico implementa compatibilidad con GnuPG a través de los plugins sylpheed-claws-pgpinline-plugin y sylpheed-claws-pgpmime. No es un cliente muy conocido, pero es bastante usado entre los más frikis de Linux.

- Mutt (*NIX): Se trata de un cliente para consola (osea, que no tiene modo gráfico) que soporta de forma nativa soporte para GnuPG. Si sylpheed es un cliente para frikis, éste es para mega-frikis de libro. xD



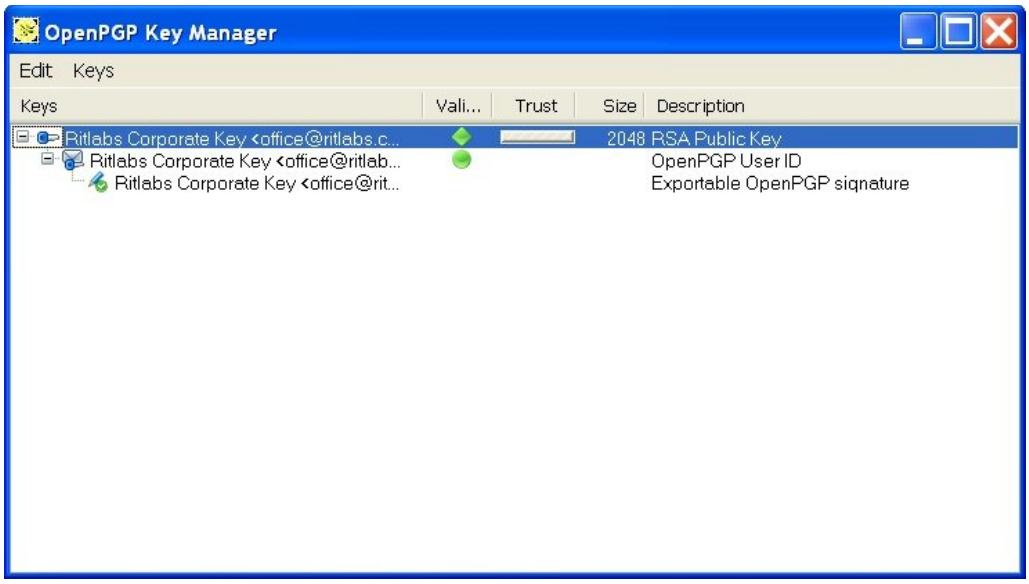
- Outlook Express (Windows): Se trata del cliente por defecto en los sistemas Windows, del mismo desarrollador (Microsoft), y no creo que haya mucha gente por aquí que no lo conozca. El instalador de PGP nos ofrece durante el proceso de instalación la opción de instalar un plugin para Outlook Express que permite integrar soporte para el mismo.

Esto que voy a decir ahora es una opinión persona, y como tal es gratis: Outlook Express (y Outlook) son seguramente los clientes de correo más usados, como suele ocurrir con todo software de la multinacional Microsoft. Así mismo -y aún a riesgo de parecer un talibán del software libre- os diré que es uno de los peores clientes de correo: sus fallos de seguridad son legendarios y han "ayudado" a hacer famosos a buen número de gusanos de correo.

¿Queréis un consejo? No uséis Outlook, hay muchas otras mejores opciones para sistemas Windows, y no necesariamente Software Libre.

- Eudora (Windows): Uno de los más veteranos clientes de correo para Windows. Al igual que ocurría con Outlook Express, PGP nos ofrece durante su instalación un plugin para integrar soporte para PGP en Eudora. Personalmente, lo probé y no me terminó de convencer, pero al igual que ocurre con Evolution, tiene grandes incondicionales. Es importante tener en cuenta que este software es de pago, por lo que deberéis abonar la correspondiente licencia para poder instalarlo (o probar una versión de evaluación).

- The Bat! (Windows): Este cliente de correo empieza a ser cada vez más conocido, y es -en mi opinión, como siempre- una de las mejores opciones dentro del software no libre, si no la mejor. Implementa compatibilidad con OpenPGP a través de diversas versiones de PGP y GnuPG, así una implementación propia empotrada en el software que permite usar cifrado de correo OpenPGP sin necesidad de instalar software adicional. Esa implementación propia posee la mayoría de las características de PGP y GnuPG, así como interfaz gráfica propia.



Todos estos clientes de correo se configura y usan de una manera muy similar (excepto Mutt, que es un "caso aparte"), así que no creo que nadie tenga ningún problema con ellos. Y si alguno de vosotros lo tiene, sólo tenéis que pasaros por el foro y preguntar, estoy seguro que alguien os ayudará encantado (hasta con Mutt, que me sé de alguno que lo usa :-P).

OpenPGP con cualquier MUA

Es bastante probable que en un momento dado necesitemos una manera de utilizar OpenPGP "universal". Por ejemplo, podemos estar usando un cliente que no tenga soporte para OpenPGP, o que lo tenga mediante complicados plugins o scripts; o puede que estemos usando un cliente webmail y hayamos recibido un correo cifrado y/o firmado que queramos descifrar y/o verificar.

Sea cual sea el caso, es útil tener a mano una forma general de usar OpenPGP para correo electrónico. Para ello vamos a usar las capacidades de cifrado de texto plano de GnuPG, KGPG y PGP. Para este ejemplo voy a mandar un correo cifrado desde mi cuenta de gmail mediante la interfaz webmail (no suelo utilizar gmail, y cuando lo hago es a través de Thunderbird).

Lo primero, vamos a escribir un texto (yo usaré mi queridísimo Vim :-P).

```
Hola, esto es una prueba...

Wadalbertia über alles!
```

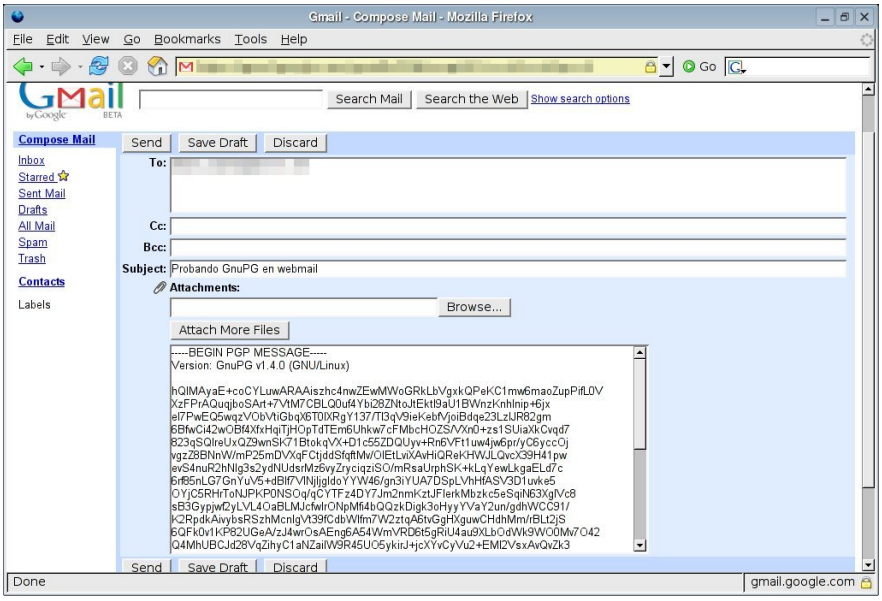
Ahora lo cifraremos con GnuPG...

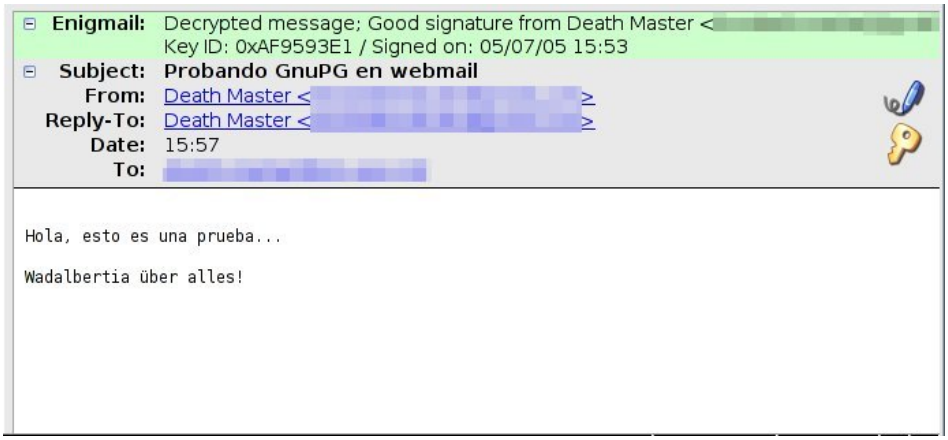
```
master@blingdenstone:~$ gpg --armor --recipient 0xaf9593e1 --sign --encrypt-files texto.txt

You need a passphrase to unlock the secret key for
user: "Death Master <correo_electrónico>"
4096-bit RSA key, ID AF9593E1, created 2003-07-21

master@blingdenstone:~$
```

Con lo que hemos obtenido un fichero texto.txt.asc que contiene:

```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.0 (GNU/Linux)

hQIMAyaE+coCYLuwARAAiszhc4nwZEwMWoGRkLbVgxkQPeKC1mw6maoZupPifL0V
{...}

-----END PGP MESSAGE-----
```

Copiamos el texto, lo pegamos en la ventana de composición de mensaje del webmail, y lo enviamos. Ahora comprobamos que ha llegado, se descifra y la firma se verifica correctamente. Consultando la consola de enigmail vemos que efectivamente así es:






```
enigmail> /usr/bin/gpg --charset utf8 --batch --no-tty --status-fd 2 -d --passph
rase-fd 0 --no-use-agent
gpg: encrypted with 4096-bit RSA key, ID 0260BBB0, created 2003-07-21
      "Death Master <correo_electrónico>"
gpg: Signature made Sat May  7 15:53:05 2005 CEST using RSA key ID AF9593E1
gpg: Good signature from "Death Master <correo_electrónico>"

{...}

enigmail> /usr/bin/gpg --charset utf8 --batch --no-tty --status-fd 2 --with-colo
ns --list-keys 6F6F3938AF9593E1
```

Mediante este método podemos transmitir mensajes cifrados por cualquier medio que permita el envío de texto, y no únicamente limitarnos al correo electrónico... pensad en otros ejemplos de comunicación con texto: IRC, mensajería instantánea... pero también existen formas de cifrar automáticamente en estos medios. Ya hablaremos de ello a su debido tiempo... ;-)

Ejemplo práctico

¿Realmente es tan necesario el uso de criptografía? ¿Son tan inseguros nuestros correos electrónicos? En realidad esto es como todo: tiene tanta importancia como tú quieras darle. No veo ningún problema en mandarle a un amigo un correo sin cifrar en el que le dices que habéis quedado a las 22.00 en la estación de metro; pero en cambio me preocupa sobremanera cómo enviar a ese mismo amigo las claves de acceso a la base de datos del foro, y quien dice claves de acceso dice número de cuentas, documentos importantes...

Por eso, vamos a ver un ejemplo de lo fácil que resulta espiar el correo electrónico. Para ello voy a usar el magnífico sniffer Ethereal (http://www.ethereal.com/) para capturar el tráfico de diversos correos que voy a enviarme a mí mismo. En este caso voy a capturar el tráfico desde el mismo ordenador que lo genera, pero sería trivial realizarlo desde cualquier otro ordenador de la misma red (con técnicas como ARP Spoofing, por ejemplo) o cualquier ordenador conectado a una red por la que pase el correo electrónico antes de llegar a su destino.

> *¡¡IMPORTANTE!! Antes de nada, debe quedar MUY clara una cosa: yo voy a utilizar cuentas de correo que son de mi propiedad por lo que no hay ningún problema. Pero en caso de que no fueran mías sí lo habría, pues espiar el correo electrónico es un DELITO MUY GRAVE.*
>
> *Cada cual es responsable de sus propios actos.*

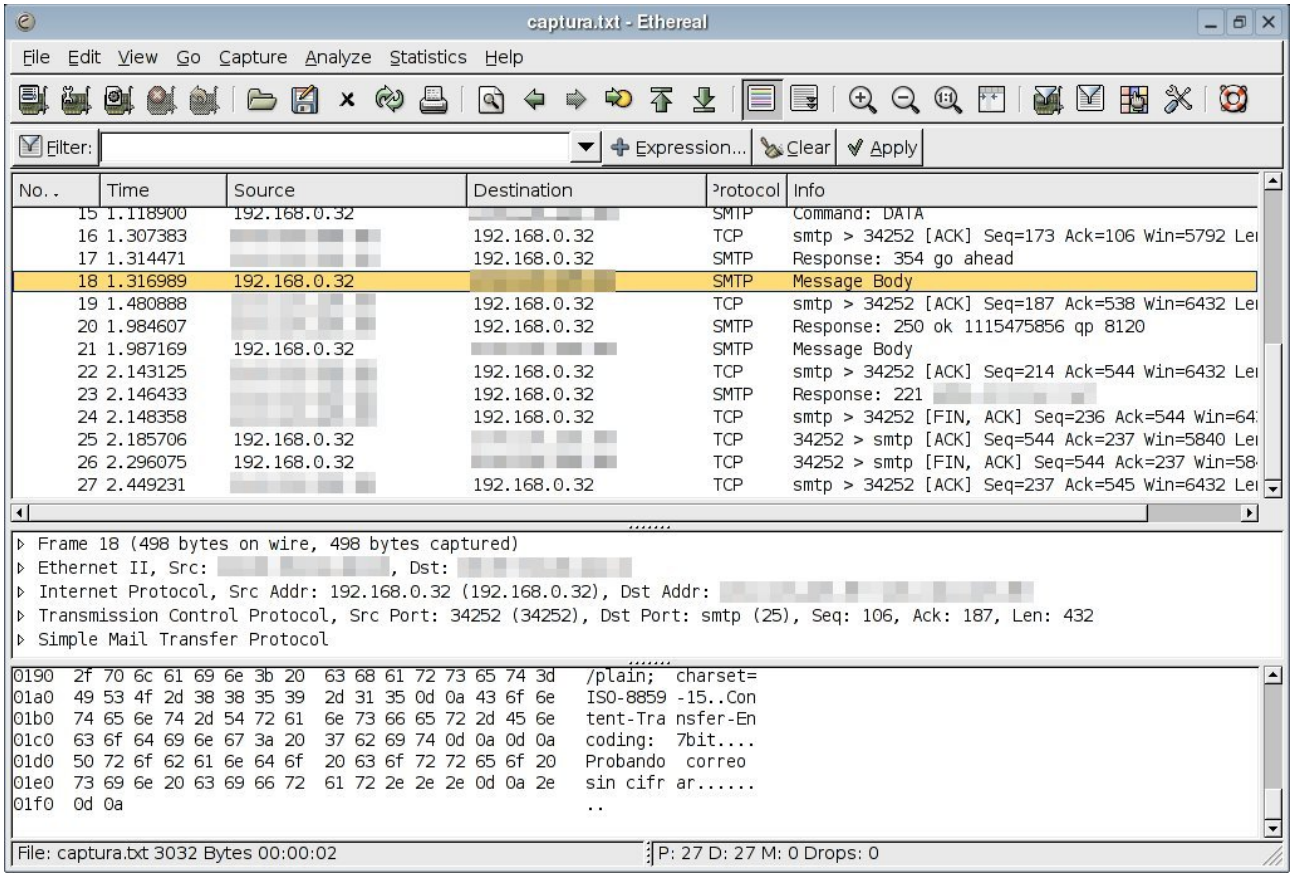
Para nuestro primer ejemplo voy a mandar un correo electrónico sin cifrar desde una cuenta con correo saliente SMTP hasta una cuenta con correo entrante POPS (con SSL). El camino que seguirá el correo será:

Origen -> Access Point -> Router -> {Internet} -> Servidor SMTP -> {Internet} -> Servidor POPS -> {Internet} -> Router -> Access Point -> Destino

Explicaré brevemente la leyenda utilizada: en verde está marcado el trayecto durante el cual el correo es completamente legible para cualquiera interesado en él; en amarillo está marcado el trayecto durante el cual no sabemos el grado de seguridad del correo (aunque posiblemente sea bajo o nulo); y en rojo está marcado el trayecto durante el cual el correo es ilegible para cualquiera (excepto para nosotros en el destino) y es por tanto seguro.

Al usar correo SMTP sin cifrar, el correo hasta llegar al servidor viaja como una postal, completamente visible. Desde el servidor SMTP hasta el servidor POPS no conocemos el grado de seguridad que mantienen las conexiones entre los nodos, pero muy posiblemente sea nulo. Desde el servidor POPS hasta nosotros, la conexión viaja cifrada con SSL por lo que podemos considerar que el correo está seguro.

En la primera flecha, la conexión entre el origen y el AP, es donde voy a capturar el tráfico. Observando el log generado por Ethereal encontramos cosas interesantes:



- Comandos del servidor: EHLO [192.168.0.32]

- Cabeceras del correo: User-Agent: Debian Thunderbird 1.0.2 (X11/20050331)

- El propio cuerpo del correo: Probando correo sin cifrar...

Como vemos, el correo entero está "al desnudo" con este envío. Pero podría haber sido peor... por ejemplo si el destinatario utilizara POP sin cifrado para la recepción del mensaje:

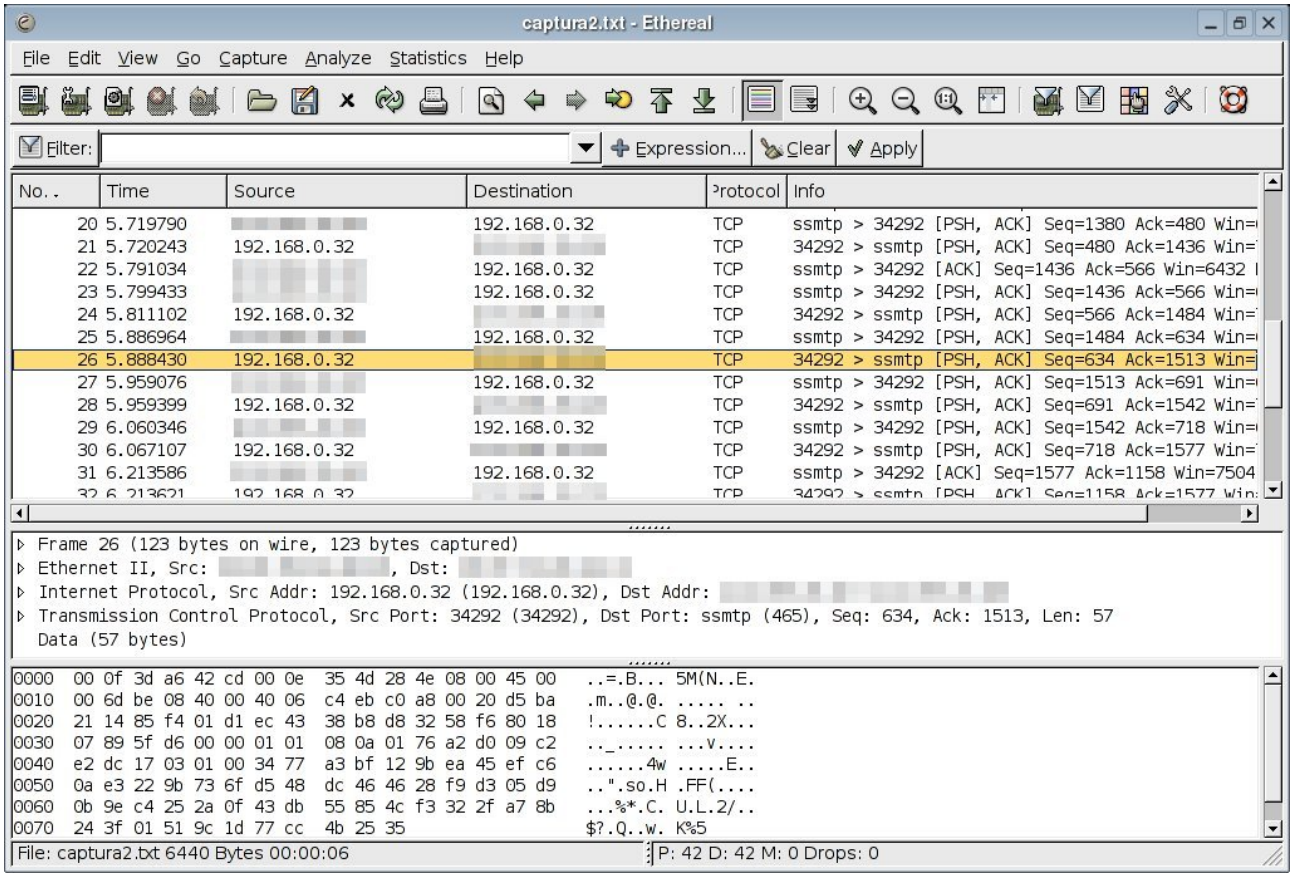
Origen -> Access Point -> Router -> {Internet} -> Servidor SMTP -> {Internet} -> Servidor POP -> {Internet} -> Router -> Access Point -> Destino

Lo que en la práctica significa una postal desde su origen a su destino.

Ahora probaremos a enviar un correo cifrado electrónico sin cifrar desde una cuenta que usa SMTPS hasta una cuenta con correo entrante POPS (ambos con SSL). El esquema de la conexión sería este:

Origen -> Access Point -> Router -> {Internet} -> Servidor SMTPS -> {Internet} -> Servidor POPS -> {Internet} -> Router -> Access Point -> Destino

Al capturar con Ethereal en el mismo punto que en el ejemplo anterior, podemos ver que solamente capturamos un galimatías sin sentido. Pero existe aún un punto flaco, en el intervalo entre el servidor SMTPS y el servidor POPS. De hecho, si el correo fuera enviado a una cuenta que no utilizara SSL para recibir correo, el esquema sería el siguiente:



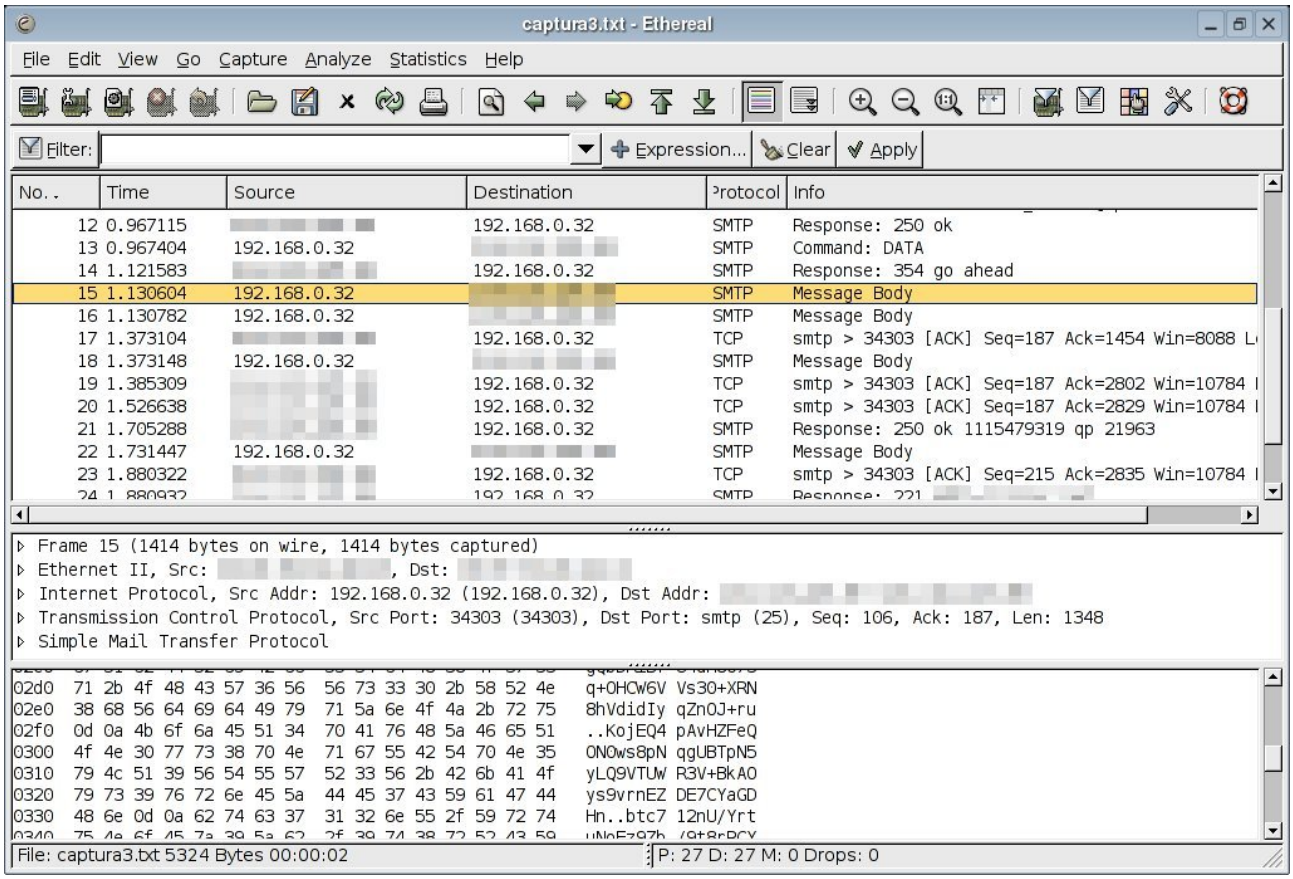
Origen -> Access Point -> Router -> {Internet} -> Servidor SMTPS -> {Internet} -> Servidor POP -> {Internet} -> Router -> Access Point -> Destino

Como vemos, SSL ofrece seguridad, pero solamente dentro de su "jurisdicción", es decir, que más allá de la conexión entre cliente y servidor, la cosa queda a merced de los medios que conecten los servidores.

Ahora probaremos a mandar un correo electrónico cifrado con PGP desde una cuenta con correo saliente SMTP hasta una cuenta con correo entrante SMTP (ambas sin SSL). El esquema es el siguiente:

Origen -> Access Point -> Router -> {Internet} -> Servidor SMTP -> {Internet} -> Servidor POP -> {Internet} -> Router -> Access Point -> Destino

Capturando con Ethereal podemos observar que únicamente podemos cotillear las cabeceras del correo, los comandos del servidor y poco más, pero que el cuerpo del mensaje aparece cifrado en forma de armadura PGP, y de ahí no podremos sacar nada. Como veréis, hemos eliminado hasta la zona muerta que cubría la conexión entre los servidores de correo y que hasta ahora no habíamos podido controlar por ningún medio. Además, al enviar correos cifrados con PGP, aunque el destinatario utilice servidores sin conexiones seguras, el correo seguirá siendo ilegible.



¿Existe alguna opción más segura? Sí, cifrar correos con PGP y usar servidores con conexiones SSL, de forma que ni tan siquiera las cabeceras y los comandos del servidor podrán ser vistos mediante sniffers.

Espero que estos inocentes ejemplos os hayan hecho pensar en la importancia de utilizar los medios que están a nuestro alcance para aumentar nuestra seguridad a la hora de utilizar las funcionalidades que las nuevas tecnologías nos brindan.

Con esto terminamos el artículo de este número. Como siempre, si existen dudas y/o problemas con cualquier práctica o elemento teórico, no tenéis más que hacernos una visita en el foro, donde os recibiremos encantados. Os espero en el próximo número, cuya temática... es una sorpresa. :-P

4- OpenPGP de andar por casa

(Publicado en "PC Paso a Paso: Los cuadernos de Hack X Crack" número 30, 17 de Agosto de 2005)

Aquí estamos una vez más, en el Taller de Criptografía, dispuestos a dar unos cuantos pasos más en el largo pero gratificante camino del aprendizaje de la criptografía. Antes de nada quiero agradecer todos los correos y mensajes privados del foro que he recibido con comentarios (todos positivos) acerca del Taller. Es muy gratificante saber que a la gente le resulta útil y ameno. Gracias a todos los que de una forma u otra me lo han hecho saber.

Hagamos recopilación de lo que hemos visto hasta ahora: en los dos primeros artículos aprendimos los fundamentos criptográficos y matemáticos del sistema OpenPGP, y realizamos una primera toma de contacto con sus dos principales baluartes: PGP y GnuPG. Fueron los artículos con más carga teórica (o al menos, no práctica). El tercer artículo supuso un gran cambio en este aspecto, pues pasamos a utilizar todo lo que ya sabíamos sobre el sistema en una situación cotidiana: la gestión del correo electrónico. Sé por los mensajes que he recibido, que ha sido éste último el que más os ha gustado -con alguna excepción- así que estáis de suerte porque el presente artículo tiene un espíritu muy similar.

En primer lugar, y al igual que hicimos con el ataque chino a SHA-1 (del que, por cierto, ya se ha publicado el ataque completo -sólo apto para los más valientes...- aquí: http://cryptome.org/wang_sha1_v2.zip), vamos a comentar una noticia de actualidad que nos enseñará (o eso espero) una valiosa lección sobre seguridad. En segundo lugar vamos a mejorar (sí, mejorar) GnuPG con nuevas posibilidades. Por último, vamos a aplicar todo lo que sabemos a otro aspecto de nuestra "vida cotidiana" en Internet: la mensajería instantánea. Empezamos...

El eslabón más débil

Cuando hablamos de la elección de un passphrase para vuestra clave privada, recordaréis que hice especial énfasis en la importancia de la robustez del mismo: de nada me sirve una clave de 4096 bits si el passpharse es el nombre de mi mascota... una clave privada es relativamente sencilla de obtener una vez se obtiene acceso de alguna forma al equipo que la almacena; y llegados a ese punto, sólo el passphrase separa al atacante de nuestros secretos.

Hay una metáfora en el mundillo de la criptografía que se repite hasta la saciedad: una cadena siempre se rompe por su eslabón más débil. Pensad en ello: ¿por dónde se rompen las cadenas o pulseras? ;-)

Llevando esta metáfora al campo de la seguridad informática nos topamos con una realidad difícil de esquivar: casi siempre hay un eslabón "débil" que poder romper. Se ha comprobado en multitud de ocasiones: servidores con cortafuegos inexpugnables que son comprometidos por un servicio vulnerable, intranets comprometidas por VPN's mal protegidas...

> *Hace muy poco pudimos comprobar una vez más el cumplimiento de esta máxima en el sistema de foros phpBB: un fallo de la versión 2.0.15 -un viejo fallo que fue correctamente corregido en 2.0.11- que permite inyectar código PHP mediante la función system()... llegando a comprometer un servidor completo a través del software phpBB.*

¿Que qué tiene todo esto que ver con la criptografía? Mucho más de lo que a priori pudiera parecer. Os comento una noticia que he conocido a través del boletín ENIGMA (http://www.ugr.es/~aquiran/cripto/enigma.htm):

Hace un año la policía italiana confiscó el servidor de un colectivo llamado Austistici/Inventati como parte de una investigación. Tras la devolución del servidor, éste fue puesto en marcha de nuevo y siguió siendo utilizado para dar servicios como páginas web, correo electrónico, listas de correo... pero hace unas semanas, el 21 de Junio, el citado colectivo descubrió que la policía había instalado una puerta trasera para espiar las comunicaciones que fluían a través de su máquina. Y no sólo eso, sino que las comunicaciones cifradas a través de SSL, en las que la gente acostumbra a confiar de forma ciega, también fueron comprometidas: se accedió a la clave privada del certificado y se instaló un sniffer que interceptó y descifró el tráfico cifrado.

> *SSL (Secure Socket Layer) es un protocolo de cifrado a nivel de socket que se implementa como una capa en la propia pila de comunicaciones del sistema (inmediatamente antes de la capa de aplicación). No es el momento de detallar qué es y cómo funciona (todo llegará)... pero sí es importante tomar conciencia de que es algo que utilizamos muy a menudo (cuando leemos el correo, cuando consultamos nuestra cuenta del banco en Internet...) y de gran importancia.*

Como vemos, un sistema en teoría completamente seguro como SSL fue vulnerado a través de su "eslabón débil": el acceso a la clave privada. Tomemos todos nota y aprendamos esta lección en la piel ajena, que siempre escuece menos...

A mí me gustan con curvas... elípticas

Ya conocemos el estándar OpenPGP (RFC #2440: ftp://ftp.rfc-editor.org/in-notes/rfc2440.txt) así como el OpenPGP/MIME (RFC #3156: ftp://ftp.rfc-editor.org/in-notes/rfc3156.txt). También conocemos los dos grandes algoritmos de cifrado asimétrico: RSA y DH/DSS. Hemos trabajado con ellos, visto sus algoritmos... ¡y hasta los hemos comprobado matemáticamente!

Pero hay otra valiosa lección en la seguridad informática: TODO es mejorable.

RSA y DH fueron los pioneros de lo que hoy se ha convertido en el estándar de cifrado: el sistema de clave pública. No fueron los únicos, existen otros sistemas como RW contemporáneos con las dos estrellas de OpenPGP, pero muy similares en planteamiento. No obstante, la criptología como cualquier otra ciencia evoluciona con el tiempo, y lo que hoy damos por seguro puede no serlo en un futuro (buscad por la red de redes acerca del algoritmo de Shor). Pero si el criptoanálisis evoluciona, también lo hace la criptografía.

> *El algoritmo de Shor (http://en.wikipedia.org/wiki/Shor's_algorithm) es un algoritmo cuántico que permite factorizar grandes números de forma fácil: su complejidad asintótica es de O((log N)^3). Si tenéis conocimientos de estudio de complejidad algorítmica sabréis que se trata de una complejidad bastante aceptable; y si recordáis lo comentado en el primer artículo, sabréis que el algoritmo RSA basa su fuerza en la dificultad para factorizar grandes números... y no es el único, pues RW se basa en el mismo principio matemático.*
>
> *Aún así podemos respirar tranquilos, dado que al ser un algoritmo cuántico requiere de un computador cuántico para ser implementado. Aunque esos computadores aún son cosa del futuro, ya se han realizado pruebas... y de hecho el algoritmo de Shor ha sido probado y demostrado por IBM en 2001: http://domino.research.ibm.com/comm/pr.nsf/pages/news.20011219_quantum.html.*
>
> *Quizá todo esto os suene un poco futurista... pero sólo es cuestión de tiempo.*

Sin llegar a hablar de criptografía cuántica (que por cierto, es algo real y que existe hoy en día, no una entelequia futurista), existen sistemas y algoritmos más modernos que suponen un gran aumento de seguridad respecto a los "tradicionales" RSA y DH. Uno de los más importantes, si no el que más, es la criptografía de curvas elípticas.

¿En qué consiste el sistema de curvas elípticas? Simplemente se trata de cambiar el marco de trabajo de un algoritmo del tradicional grupo multiplicativo de un cuerpo finito primo a un sistema de curvas elípticas.

> *El sistema de curvas elípticas fue propuesto por primera vez por V.S. Miller en 1986 con su artículo [Mil86] Use of elliptic curve in cryptography. In Advances in Cryptology.*
>
> *Para saber más sobre las curvas elípticas, recomiendo visitar este enlace en la Wikipedia (la enciclopedia libre): http://en.wikipedia.org/wiki/Elliptic_curve. También existe una versión traducida al castellano en: http://es.wikipedia.org/wiki/Curvas_el%C3%ADpticas.*
>
> *Para los que deseen profundizar en los fundamentos matemáticos de la criptografía de curvas elípticas os recomiendo visitar este enlace: http://en.wikipedia.org/wiki/Elliptic_curve_cryptography.*

Aunque el sistema de curvas elípticas ha sido implementado con éxito sobre varios algoritmos, ha demostrado ser particularmente efectivo en algoritmos basados en el problema del logaritmo discreto... como por ejemplo DH. Y creo que ya sabéis a dónde nos lleva esto: a implementar compatibilidad con cifrado de curvas elípticas en GnuPG.

> *¿Y porqué no en PGP? Porque PGP es software propietario y NO podemos modificar ni una línea sin incurrir en un delito. En cambio, GnuPG es software libre y podemos modificarlo tanto como queramos (siempre que respetemos la licencia GPL).*
>
> *Por cierto, aprovecho para decir que estoy MUY cabreado con la actitud de PGP que, por primera vez con su nueva versión 9, NO ha publicado una versión gratuita (freeware) sino una trial con límite de tiempo. :-(*

Esta modificación de GnuPG es posible gracias al proyecto ECCGnuPG, que es el trabajo de final de carrera de unos estudiantes de la Universidad de Lleida: Ramiro Moreno Chiral (tocayo mío), Sergi Blanch i Torné y Mikael Mylnikov. La página oficial del proyecto es ésta: http://alumnes.eps.udl.es/~d4372211/index.es.html.

Como en el segundo artículo ya detallamos la compilación de GnuPG, ciertos detalles del proceso de compilación los obviaré. Lo primero es bajar la última versión de GnuPG (en el momento de escribir estas líneas, la 1.4.1): ftp://ftp.gnupg.org/gcrypt/gnupg/gnupg-1.4.1.tar.gz. Ahora debemos bajar el parche (en fichero diff) de ECCGnuPG para esa versión: http://alumnes.eps.udl.es/~d4372211/src/gnupg-1.4.1-ecc0.1.6.diff.bz2.

Primero (y así practicamos un poco) vamos a comprobar que tenemos todos los mismos ficheros...

```
master@blingdenstone:~$ ls -l gnupg-1.4.1.tar.gz
-rw-r--r--  1 master master 4059170 Jul  3 20:12 gnupg-1.4.1.tar.gz
master@blingdenstone:~$ md5sum gnupg-1.4.1.tar.gz
1cc77c6943baaa711222e954bbd785e5  gnupg-1.4.1.tar.gz
master@blingdenstone:~$ sha1sum gnupg-1.4.1.tar.gz
f8e982d5e811341a854ca9c15feda7d5aba6e09a  gnupg-1.4.1.tar.gz
master@blingdenstone:~$ ls -l gnupg-1.4.1-ecc0.1.6.diff.bz2
-rw-r--r--  1 master master 17603 Jul  3 20:04 gnupg-1.4.1-ecc0.1.6.diff.bz2
master@blingdenstone:~$ md5sum gnupg-1.4.1-ecc0.1.6.diff.bz2
a77a9fd556337faea3b5a6214b8a3a1c  gnupg-1.4.1-ecc0.1.6.diff.bz2
master@blingdenstone:~$ sha1sum gnupg-1.4.1-ecc0.1.6.diff.bz2
1d484fafd47fa31eae20c22bfc5be3c6ba6f4381  gnupg-1.4.1-ecc0.1.6.diff.bz2
master@blingdenstone:~$
```

Bien, ahora descomprimimos GnuPG y nos posicionamos en el directorio con las fuentes.

```
master@blingdenstone:~$ tar xvfz gnupg-1.4.1.tar.gz
{...}
master@blingdenstone:~$ cd gnupg-1.4.1
master@blingdenstone:~/gnupg-1.4.1$
```

Ahora es el momento de parchear el código fuente de GnuPG con ECCGnuPG. Este parche se aplica de igual forma que cualquier otro fichero diff (como por ejemplo los parches del kernel de Linux).

```
master@blingdenstone:~/gnupg-1.4.1$ bzcat ../gnupg-1.4.1-ecc0.1.6.diff.bz2 | patch -p1
patching file cipher/ecc.c
patching file cipher/ecc.h
patching file cipher/Makefile.am
patching file cipher/Makefile.in
patching file cipher/pubkey.c
patching file configure
patching file g10/getkey.c
patching file g10/keygen.c
patching file g10/keyid.c
patching file g10/mainproc.c
patching file g10/misc.c
patching file g10/seskey.c
patching file include/cipher.h
master@blingdenstone:~/gnupg-1.4.1$
```

El comando patch es uno de los más útiles para modificar código fuente en sistemas Unix. Si quieres saber más de él, consulta la página del manual: man patch.

Si no ha habido ningún problema, deberíais ver exactamente las líneas anteriores. Ahora es el momento de generar el makefile:

```
master@blingdenstone:~/gnupg-1.4.1$ ./configure
{...}
config.status: creating po/POTFILES
config.status: creating po/Makefile
config.status: executing g10defs.h commands
g10defs.h created

        Configured for: GNU/Linux (i686-pc-linux-gnu)

master@blingdenstone:~/gnupg-1.4.1$
```

Pasamos a compilar el software...

```
master@blingdenstone:~/gnupg-1.4.1$ make
{...}
make[2]: Leaving directory `/home/master/gnupg-1.4.1/checks'
make[2]: Entering directory `/home/master/gnupg-1.4.1'
make[2]: Leaving directory `/home/master/gnupg-1.4.1'
make[1]: Leaving directory `/home/master/gnupg-1.4.1'
master@blingdenstone:~/gnupg-1.4.1$
```

Y por último lo instalamos para sobrescribir la versión existente de GnuPG. Este último paso yo no lo haré, pero vosotros podéis hacerlo si queréis.

```
master@blingdenstone:~/gnupg-1.4.1$ make install
{...}
master@blingdenstone:~/gnupg-1.4.1$
```

Ahora veamos los algoritmos soportados por un GnuPG normal y corriente...

```
master@blingdenstone:~$ gpg --version
gpg (GnuPG) 1.4.1
Copyright (C) 2005 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.

Home: ~/.gnupg
Supported algorithms:
Pubkey: RSA, RSA-E, RSA-S, ELG-E, DSA
Cipher: 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH
Hash: MD5, SHA1, RIPEMD160, SHA256, SHA384, SHA512
Compression: Uncompressed, ZIP, ZLIB, BZIP2
master@blingdenstone:~$
```

Y los soportados por la versión modificada con ECCGnuPG:

```
master@blingdenstone:~$ ./gnupg-1.4.1/g10/gpg --version
gpg (GnuPG) 1.4.1-ecc0.1.6
Copyright (C) 2005 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.

Home: ~/.gnupg
Supported algorithms:
Pubkey: RSA, RSA-E, RSA-S, ELG-E, DSA, ECC, ECELG, ECDSA
Cipher: 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH
Hash: MD5, SHA1, RIPEMD160, SHA256, SHA384, SHA512
Compression: Uncompressed, ZIP, ZLIB
master@blingdenstone:~$
```

¡Genial! Ya tenemos soporte para algoritmos basados en el sistema de curvas elípticas: ECC, ECELG, ECDSA. Y como podemos ver, el funcionamiento general de GnuPG no ha variado... crear una clave con cualquiera de estos algoritmos es tan sencillo como hacerlo para los estándar de GnuPG:

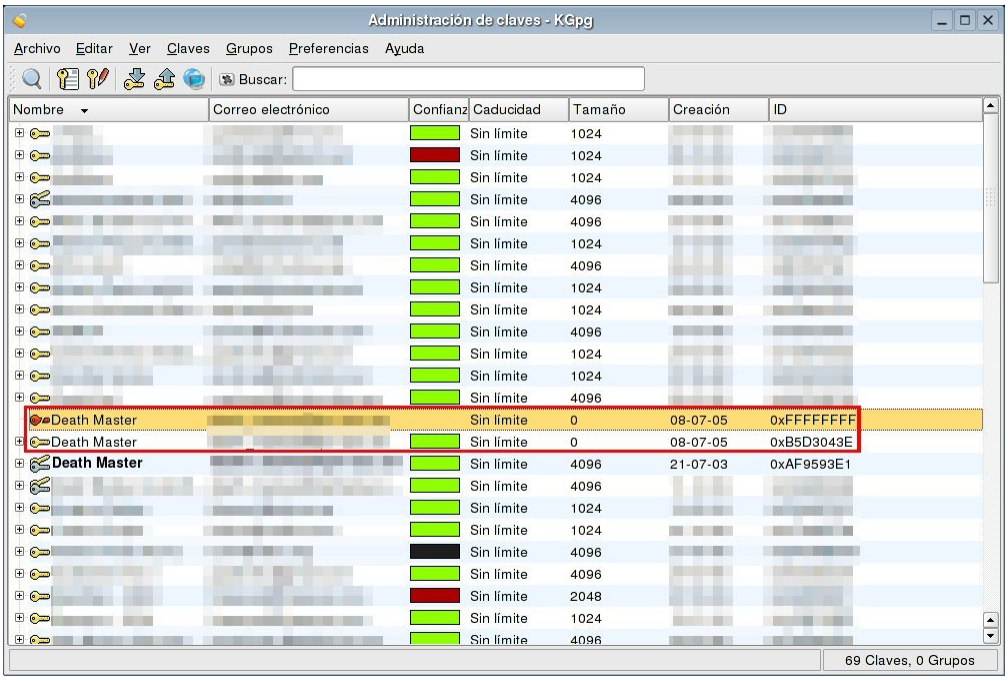
```
master@blingdenstone:~/gnupg-1.4.1/g10$ ./gpg --gen-key
gpg (GnuPG) 1.4.1-ecc0.1.6; Copyright (C) 2005 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

Please select what kind of key you want:
   (1) DSA and Elgamal (default)
   (2) DSA (sign only)
   (5) RSA (sign only)
 (103) ECC (ECElGamal & ECDSA)
Your selection?
```

Eso sí, debemos tener en cuenta que para poder utilizar el sistema ECCGnuPG, éste debe ser usado por todos las partes implicadas en el flujo de datos: no podemos pretender que nos envíen correo cifrado a nuestra clave ECC si el remitente utiliza GnuPG estándar.

Si finalizamos el proceso de creación de una nueva clave con ECC y exportamos su parte pública a una armadura ASCII, veremos que ésta tiene notables diferencias con las que estamos acostumbrados a ver:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.1 (GNU/Linux)

mQJrBELOUUtnAgkB/////////////////////////////////////////////
//////////////////////////////////8CCQH/////////////////////
////////////////////////////////////////////////////////////
/AIHUZU+uWGOHJofkpohoLaFQO6i2nJbmbMV87i0iZGO8QnhVhk5Uex+k3sWUsC9
O7G/BzVz34g9LDTx70Uf1GtQPwACCMaFjga3BATpzZ4+y2YjlbRCnGSBOQU/tSH4
KK9ga009uqFLXnfv51ko/h3BJ6L/qN4zSLPBhWpCm/l+fjHC5b1mAgkBGDkpania
O8AEXIpftCx9G9mY9URJV5tEaBevvRcnPmYsl+5ymV70JkDFULkBP60HYTU8cIai
csJAiL6Udp/RZlAAAQECCQH/////////////////////////////////////////
//pRhoeDvy+Wa3/MAUj3CaXQO7XJuImcR667b7cekThkCQIJAQ0WI6t+Kg5eTjri
rrN/QEVvJcRQU3CwEX6/8J0DiE4JBy0B4f0pbWyz2/L7QauFMkOnKH3Xfp3cqBsd
6UCPQssYAgkBuyZb4jMwj9OoxG1mkaI4X6ARDy9QBh7ot1fn+ciPAhK8MNzvyiFH
tNTl6kPKl0e73Xpdyo/jxVmfvptxkQAw2nMCCQETEo4cMY94hD3pKp4dKVD8T5/n
UJcxeS8y9TDJWkaIj19xJExw9WBmQlzPfnff7/iqnbUiflAgv92dSkfvd/PKjrQn
RGVhdGggTWFzdGVyIDxkZWF0aF9tYXN0ZXJAaHBuLXNlYy5uZXQ+iLYEE2cCABsF
AkLOUUsGCwkIBwMCAxUCAwMWAgECHgECF4AACgkQaVdf3rXTBD7iwAIIgP6H0JS+
dJHIKbA6SRinh/Ig6TcpufMjtf45ZLqJWiGdxGxsfMBjM9asCfA1ipoGJadoCmf9
6pq9qocZkP7RkcMCCQHXWajiPdBsp5JUqFEbNteyTUKTLD+NcMXUNcPn/7G4+Fog
kQYdBMUwKPIT2k/tHJp00af3q96EFNsYHj/hJ3EoIw==
=j++4
-----END PGP PUBLIC KEY BLOCK-----
```

Como vemos, la diferencia con lo que pudiera ser una clave RSA o DH es bastante llamativa (no pegaré ejemplos de estas claves a estas alturas, podéis vosotros mismos comprobarlo). Si comparamos dos claves, una RSA y otra DH, pero ambas orientadas al grupo multiplicativo, veremos que se parecen bastante. En cambio, si comparamos la clave anterior con una clave DH estándar, veremos que aunque el algoritmo es el mismo, los diferentes campos de aplicación (curvas elípticas y grupo multiplicativo de un cuerpo finito primo respectivamente) generan claves manifiestamente diferentes.



Cabe destacar que aunque con nuestro GnuPG parcheado con ECC no tendremos ningún problema para manejar estas nuevas claves, los entornos gráficos de GnuPG no están preparados para este tipo de claves, por lo que no las reconocerán y tendremos ciertos problemas con ellos, por ejemplo con KGpg y con GPA.

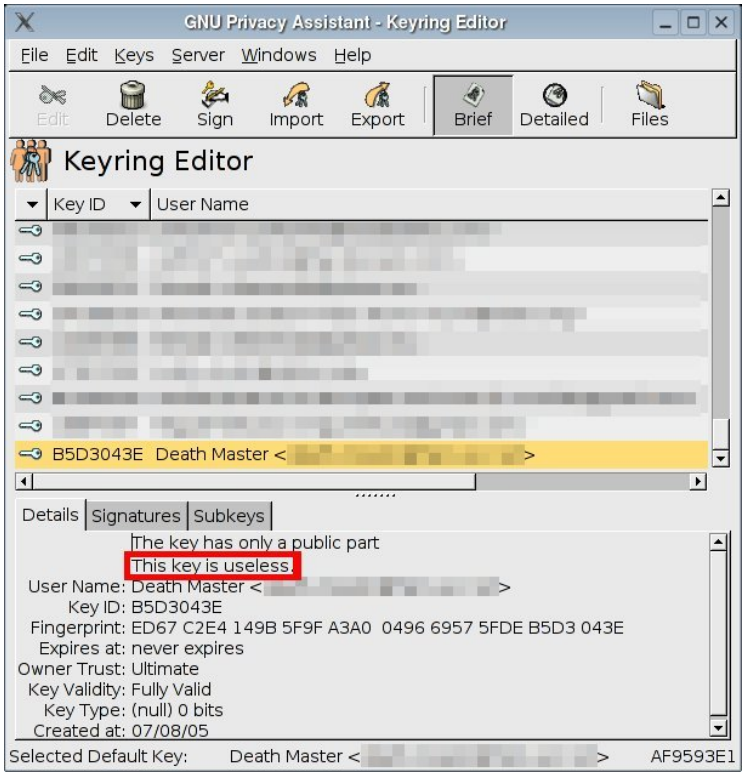
Quizá alguno de vosotros se pregunte si realmente es necesario desterrar el GnuPG estándar para utilizar ECCGnuPG. La respuesta es NO... porque es perfectamente posible que ambos coexistan en tu sistema sin que se peleen. ;-)

Si recordáis el segundo artículo, en el que hablamos de GnuPG, comenté que yo tengo en mi sistema dos versiones de GnuPG: la estable y la de desarrollo:

```
master@blingdenstone:~$ gpg --version
gpg (GnuPG) 1.4.1
{...}
master@blingdenstone:~$ gpg2 --version
gpg (GnuPG) 1.9.15
{...}
master@blingdenstone:~$
```

Pero además tengo también la versión que acabamos de compilar de ECCGnuPG:

```
master@blingdenstone:~$ eccgpg --version
gpg (GnuPG) 1.4.1-ecc0.1.6
{...}
master@blingdenstone:~$
```



Las dos versiones estándar de GnuPG están instaladas mediante el gestor de paquetes de mi distribución (APT, Advanced Packaging Tool), mientras que ECCGnuPG es la que acabamos de compilar con un enlace al directorio de binarios del sistema:

```
master@blingdenstone:~$ ls -l /usr/bin/gpg
-rwsr-xr-x  1 root root 809836 May 10 00:47 /usr/bin/gpg
master@blingdenstone:~$ ls -l /usr/bin/gpg2
-rwsr-xr-x  1 root root 544332 Apr  8 15:53 /usr/bin/gpg2
master@blingdenstone:~$ ls -l /usr/bin/eccgpg
lrwxr-xr-x  1 root root 35 Jul  8 12:09 /usr/bin/eccgpg -> /home/master/eccgnupg-1.4.1/g10/gpg
master@blingdenstone:~$
```

Para realizar un enlace simbólico sólo debemos convertirnos en usuario privilegiado y ejecutar la siguiente orden:

```
master@blingdenstone:~$ su
Password:
blingdenstone:/home/master# cd /usr/bin
blingdenstone:/usr/bin# ln -s /home/master/eccgnupg-1.4.1/g10/gpg eccgpg
blingdenstone:/usr/bin# exit
exit
master@blingdenstone:~$ eccgpg --version
gpg (GnuPG) 1.4.1-ecc0.1.6
{...}
master@blingdenstone:~$
```

Ahora, cuando queramos utilizar ECCGnuPG en un software determinado (por ejemplo, en Mozilla Thunderbird para nuestro correo electrónico) sólo deberemos indicar como ruta del ejecutable /usr/bin/eccgpg.

Os animo a que practiquéis con ECCGnuPG todas las opciones de las que ya hablamos anteriormente y me contéis vuestras experiencias en el foro. :-)

> *Por si aún hay alguien que no conozca nuestro foro -difícil, porque lo digo cada número :-P-, que no deje de hacernos una visita en http://www.hackxcrack.com/phpBB2/*

Hola guapa, ¿me das tu MSN... y tu clave PGP?

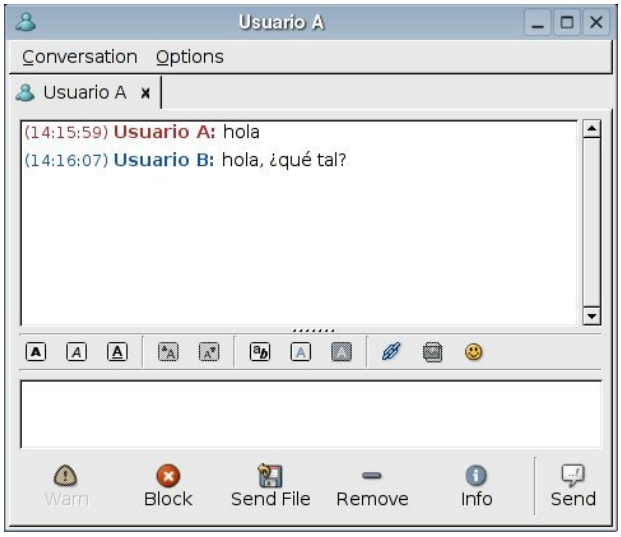
Frase extraída del afamado manual "Cómo no volver a ver a una chica en tu vida" xD. Ahora en serio... creo que a estas alturas no habrá nadie que no sepa lo que es el famoso "messenger" pero por si acaso haré una breve introducción.

En 1996 la compañía AOL (America OnLine Inc) ideó un sistema de comunicación instantánea entre sus usuarios. En 1997 introdujo una novedad que lo convirtió en el embrión de lo que hoy conocemos como mensajería instantánea: la lista de contactos. Por desgracia AIM (AOL Instant Messenger) era exclusivo para clientes de la compañía... y ahí hizo su aparición estelar el primer gran IM de la historia: ICQ. En su día tremendamente popular, y hoy en día desconocido por mucha gente (pocos nostálgicos conservamos una cuenta ICQ y seguimos haciendo login en ella), sentó las bases de lo que después sería el boom de Internet.

De la mano de Yahoo Messenger, Hotmail Messenger (conocido como MSN Messenger desde que Microsoft compró hotmail) y otros menos conocidos como Jabber o Gadu-Gadu, la mensajería instantánea se ha convertido en un fenómeno de masas muy superior a otros que, en principio, tenían más papeletas para serlo (correo electrónico, IRC, VoIP...). A algunas personas no les gusta demasiado (a un servidor, sin ir más lejos) y a otras les apasiona... pero todas corren el mismo peligro usándolo.

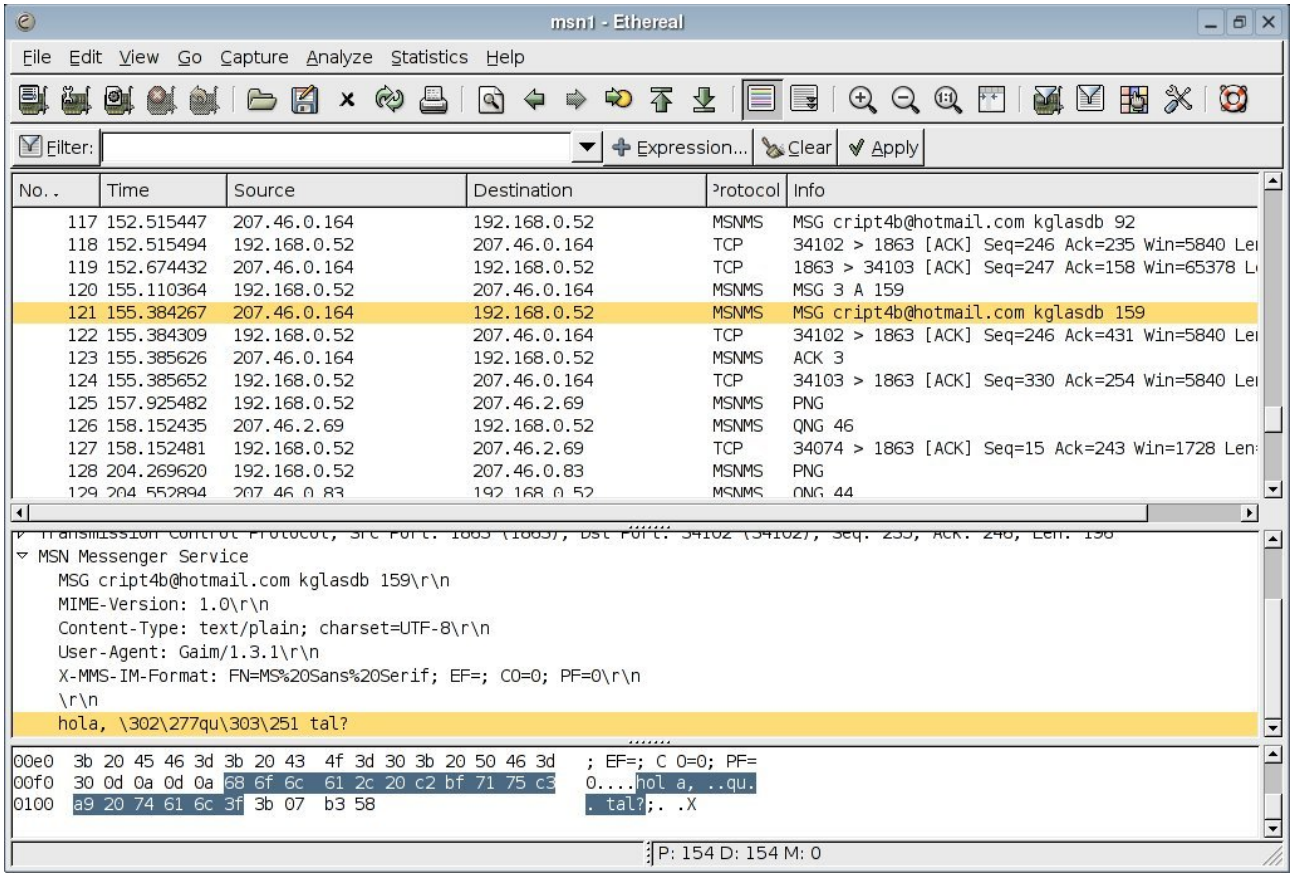
Sí, ya sé que estaréis pensando *"ya está aquí el agorero para amargarme la conversación..."* pero al igual que con el correo electrónico, creo que no se conoce un sistema a fondo hasta que no conocemos completamente sus debilidades y cómo solucionarlas. Veamos un ejemplo...

Nuestro querido e incauto Usuario A inicia sesión con su cuenta de MSN y se encuentra a su amigo del alma Usuario B, por lo que decide iniciar una conversación con él. Pero no es el único que está escuchando esa conversación...



*¡¡**IMPORTANTE!!** Antes de nada, debe quedar **MUY** clara una cosa: yo voy a utilizar cuentas de correo que he creado específicamente para el desarrollo de este artículo, y todo el tráfico capturado pertenece a **MI** red local por lo que no hay ningún problema. Pero en otro caso sí lo habría, pues espiar las conversaciones instantáneas es un **DELITO MUY GRAVE**. A algunas personas este hecho no les amedrenta (por eso aprendemos a defendernos), pero debe quedar absolutamente claro este aspecto.*

Cada cual es responsable de sus propios actos.

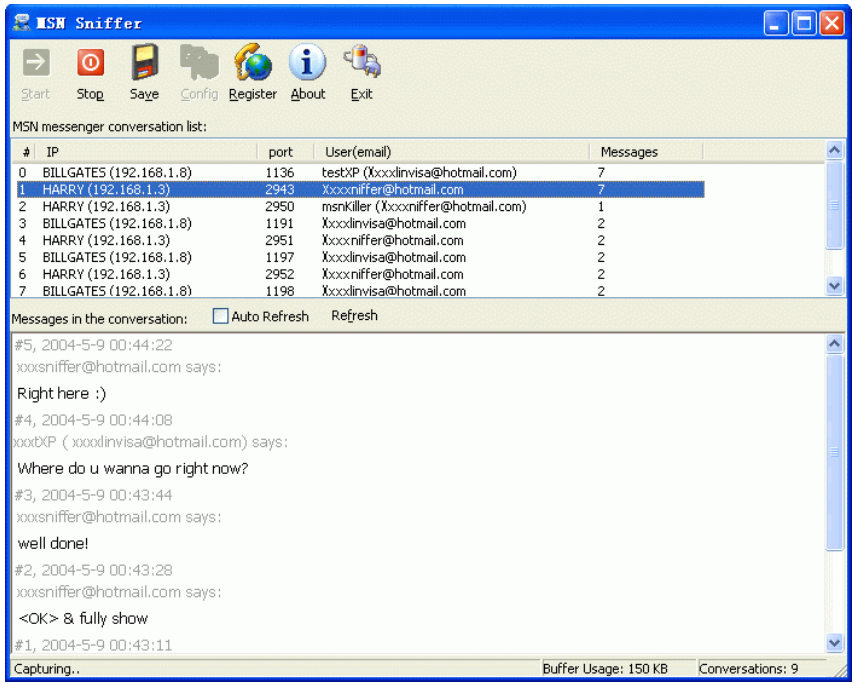


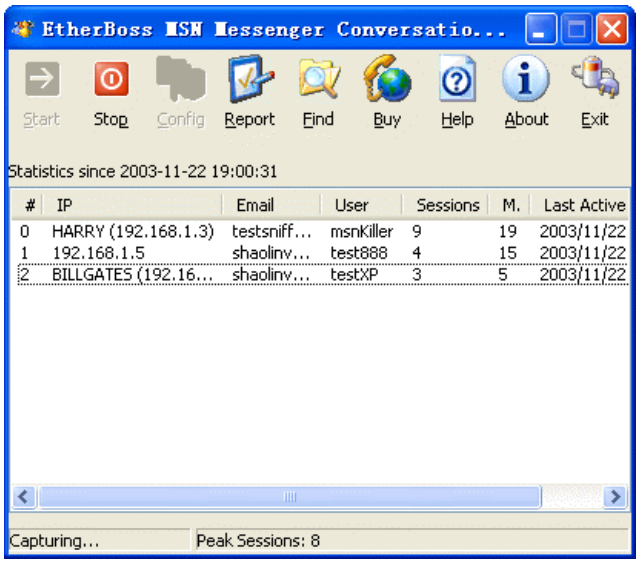
Como vemos, un hipotético atacante con acceso a la red -con técnicas que ya hemos visto muchas veces- de uno de los dos interlocutores (en este caso particular, de ambos) puede espiar la conversación con un simple sniffer y un filtro para controlar el puerto 1863.

Si bien el protocolo de MSN Messenger no es un estándar público -por lo que no se encuentra detallado en ningún RFC-, sino propietario de la multinacional Microsoft, existe en la red muchísima información acerca del mismo

Para los coleccionistas de Hack X Crack: si echáis la vista atrás... muy atrás, concretamente al número 15, encontraréis en el artículo número 9 de la serie RAW del fantástico PyC en el que destripa el protocolo de MSN Messenger MSNP7. Por desgracia, ese protocolo dejó de funcionar hace un tiempo en detrimento de sucesivas revisiones: MSNP8, MSNP9... hasta la última, la versión MSNP11 que coincide con el nuevo MSN Messenger 7. No obstante, son prácticamente idénticos en la mayoría de sus aspectos principales (el que más ha cambiado ha sido el desafío de autenticación contra el servidor)

Como vemos, la interpretación de los paquetes capturados es casi trivial, al menos en el aspecto que podría interesar a un atacante algo cotilla: la conversación en sí. Aún así existen programas que automatizan toda esa tarea para reconstruir automáticamente la conversación, como por ejemplo MSN Sniffer (http://www.effetech.com/msn-sniffer/), EtherBoss MSN Messenger Conversation Monitor & Sniffer (http://www.etherboss.com/msn-monitor/), IM Sniffer (http://www.johnytech.com/product.asp?Id=15), todos ellos para sistemas Windows; o imsniff (http://freshmeat.net/projects/imsniff/) para sistemas Unix. Y MSN Messenger no es el único sistema de mensajería instantánea que

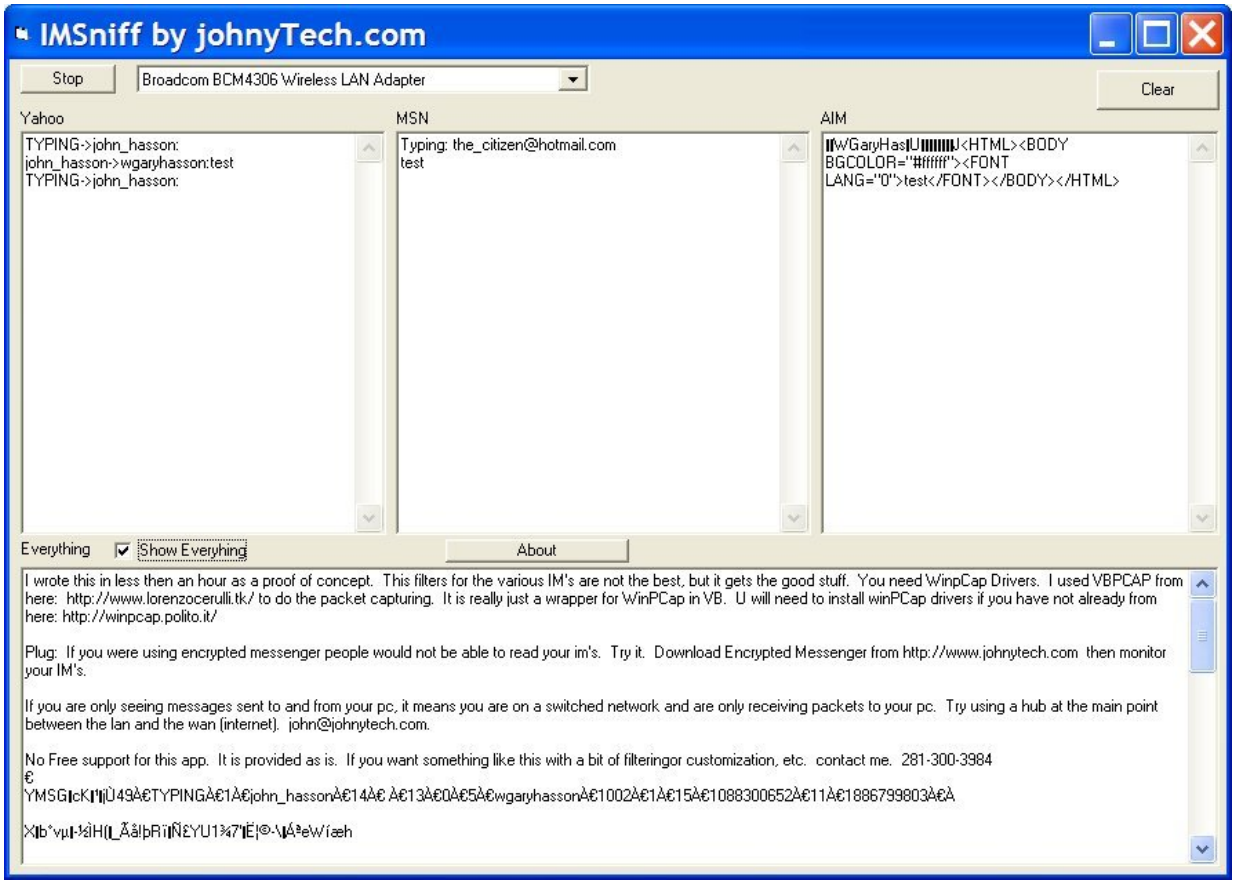




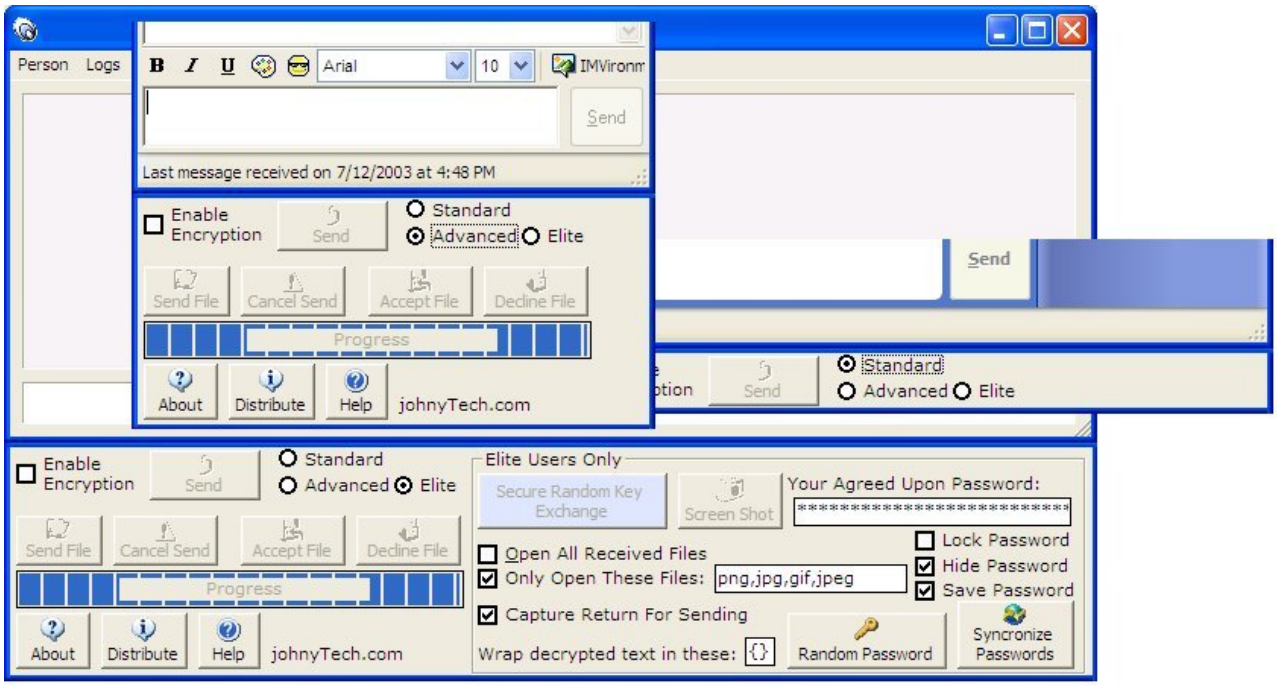
posee "espías a medida": ICQ, AOL... todos tienen los suyos. Basta una pequeña visita a Google para verificarlo

¿Y qué podemos hacer nosotros para protegernos de todos estos espías? Bueno, lo más sencillo es no usar mensajería instantánea... es broma, es broma :-P. Lo mejor -como siempre ;-)- es echar mano de la criptografía.

Existen algunos programas para cifrar las conversaciones de mensajería instantánea como Encrypted Messenger (http://www.johnytech.com/product.asp?Id=14), que permite cifrar conversaciones en múltiples protocolos de mensajería y clientes diferentes.



Esta clase de programas de cifrado están muy bien, aportan seguridad a nuestras comunicaciones y todo eso... pero su sistema de cifrado no nos es familiar (y, creedme, en los programas de cifrado es habitual encontrar puertas traseras...). A nosotros lo que nos interesa es... OpenPGP, claro. De hecho, existen algunos programas para implementar cifrado OpenPGP en MSN Messenger.



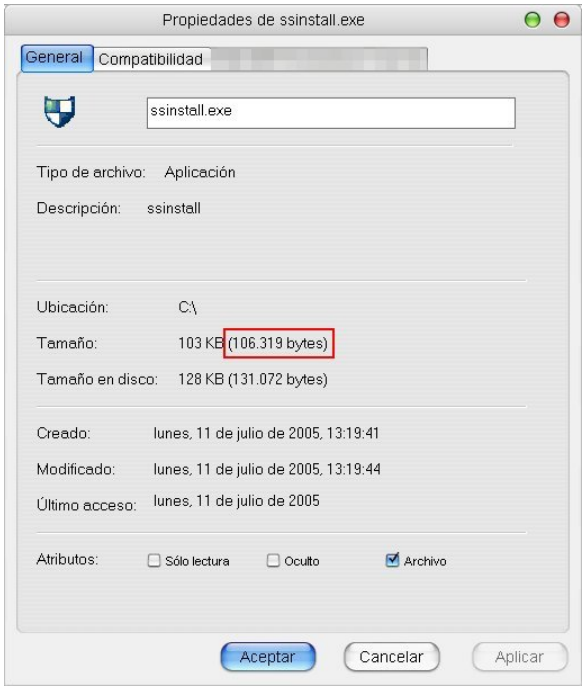
Quizá uno de los primeros programas para esta tarea fue Spyshield (http://www.commandcode.com/spyshield.html). Hoy en día está algo desfasado, pues implementa compatibilidad únicamente para MSN Messenger 4.x (y ahora andan por la versión 7...) y Windows Messenger. Además, para los usuarios de Windows presenta otra gran desventaja: únicamente funciona con un sistema OpenPGP de línea de comandos, es decir, se hace imprescindible utilizar una versión "veterana" de PGP (os recomiendo la 6.5.8) o bien usar GnuPG.

> *La versión 6.5.8 es una versión "legendaria" de PGP. ¿Por qué? Por ser la última versión de línea de comandos que tuvo PGP. A partir de la versión 7 (más tarde la 8 -la que yo uso en Windows- y la actual 9) se desechó por completo el uso de línea de comandos para PGP, integrando el software con Windows mediante su shell gráfica.*
>
> *Fueron MUCHAS las personas que, llegada la versión 7 de PGP, decidieron continuar con 6.5.8 para poder aprovechar la potencia y versatilidad de la línea de comandos. De hecho... ¿recordáis las claves "RSA Legacy" que se mantenían por razones de compatibilidad? Pues la versión 6.5.8 es el motivo.*
>
> *Aún hay gente que utiliza PGP 6.5.8 de forma habitual, si bien muchos de sus usuarios han ido migrando cada vez más al genial GnuPG, que ofrece una interfaz de línea de comandos a la vez que compatibilidad con los algoritmos más modernos de cifrado... de hecho acabáis de ver vosotros mismos cómo implementar cifrado de curvas elípticas sin renunciar a la línea de comandos.*
>
> *Podéis descargar PGP 6.5.8 (para múltiples sistemas) de aquí: http://web.mit.edu/network/pgp.html*

Si deseáis probar Spyshield, podéis bajarlo de aquí (como siempre, comprobamos los hashes del fichero descargado): http://www.commandcode.com/downloads/ssinstall.exe.

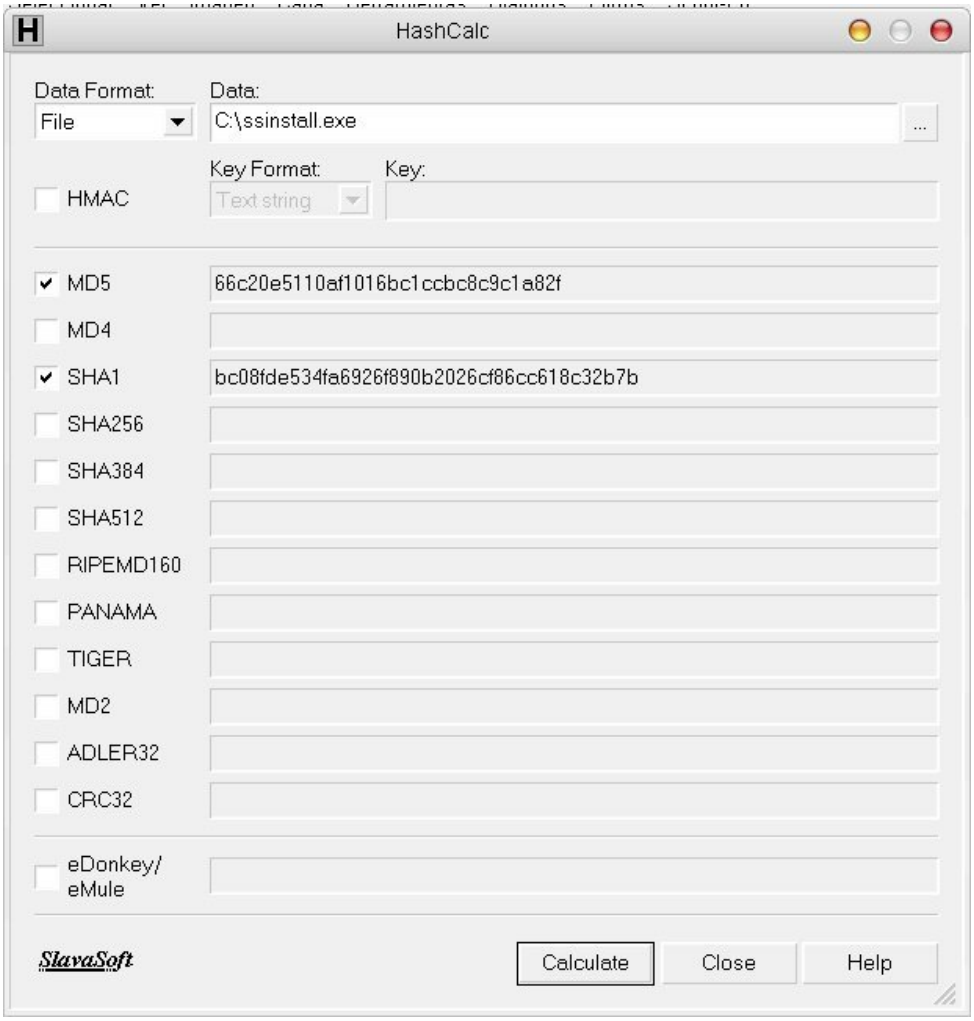
```
master@blingdenstone:~$ ls -l ssinstall.exe
-rw-r--r--  1 master master 106319 Jul  9 13:20 ssinstall.exe
master@blingdenstone:~$ md5sum ssinstall.exe
66c20e5110af1016bc1ccbc8c9c1a82f  ssinstall.exe
master@blingdenstone:~$ sha1sum ssinstall.exe
bc08fde534fa6926f890b2026cf86cc618c32b7b  ssinstall.exe
master@blingdenstone:~$
```

> *Llegados a este punto -y tras la compilación de ECCGnuPG en Linux- muchos de vosotros trabajaréis con un sistema Windows para seguir esta parte del artículo. Así pues, ¿cómo realizamos las comprobaciones de tamaño, hash MD5 y hash SHA-1 en Windows? Muy fácil.*
>
> *Para comprobar el tamaño, seleccionaremos con el botón derecho el fichero a comprobar y elegiremos la opción "Propiedades". En el campo "Tamaño" podremos ver los bytes que ocupa: 106.319 (los mismos que nos devuelve al ejecutar "ls -l ssinstall.exe".*
>
> *Para comprobar los hashes (tanto MD5 como SHA-1) vamos a utilizar un programa llamado HashCalc (que podemos descargar de http://www.slavasoft.com/hashcalc/) que sirve para -como su propio nombre indica- realizar cálculos de hashes en gran cantidad de algoritmos diferentes.*



En nuestro caso seleccionaremos MD5 y SHA-1 e indicaremos la ruta al fichero. Como podréis comprobar, los valores hash devueltos son los mismos que hemos calculado con md5sum y sha1sum en Linux.

Una vez instalado, al ejecutarlo por primera vez nos pedirá la ruta a PGP o GnuPG. Tras eso, sólo es necesario arrancar el messenger y posteriormente Spyshield.



Cifrando conversaciones de forma SIMPle

Si bien Spyshield cumple perfectamente con su cometido, está algo desfasado en cuanto al software que soporta, por lo que echaremos un vistazo a otro programa más moderno que cumple la misma función... ese software se llama SIMP (Secway Instant Messenger Privacy), y está disponible para los protocolos de mensajería de MSN, Yahoo!, ICQ y AIM. La versión Pro (http://www.secway.fr/products/simppro/home.php?PARAM=us,ie) soporta todos los protocolos citados, mientras que las versiones Lite solamente soportan uno de ellos (es decir, hay cuatro Simp Lite diferentes). Todas las versiones Lite son gratuitas para uso personal (un detalle por parte de Secway, la empresa desarrolladora del software).

Nosotros vamos a centrarnos en SIMP Lite para el protocolo MSN Messenger (http://www.secway.fr/products/simplite_msn/home.php?PARAM=us,ie).
Vamos, pues a descargar tanto el software como la firma PGP (que para algo la dan, ¿no? ;-P) y comprobamos tanto los hashes como la firma: http://www.secway.fr/products/simplite_msn/download.php?PARAM=us,ie.

```
master@blingdenstone:~$ wget http://www.secway.fr/resources/pgp/secway.asc
--14:11:38--  http://www.secway.fr/resources/pgp/secway.asc
        => `secway.asc'
Resolving www.secway.fr... 213.186.33.16
Connecting to www.secway.fr|213.186.33.16|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2,385 (2.3K) [text/plain]

100%[==========================================================
============>] 2,385        --.--K/s

14:11:44 (99.79 KB/s) - `secway.asc' saved [2385/2385]

master@blingdenstone:~$ gpg --import secway.asc
gpg: key 9C83ABF1: public key "Secway <contact@secway.fr>" imported
gpg: Total number processed: 1
gpg:               imported: 1
master@blingdenstone:~$ gpg --verify-files SimpLite-MSN-2_1_6-en.msi.sig
gpg: Signature made Fri Apr  8 18:47:53 2005 CEST using DSA key ID 9C83ABF1
gpg: Good signature from "Secway <contact@secway.fr>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 90C6 4F05 41E5 8099 E015  227B 826C 845A 9C83 ABF1
master@blingdenstone:~$ ls -l SimpLite-MSN-2_1_6-en.msi
-rw-rw-rw-  1 master master 3488768 Jul  9 13:37 SimpLite-MSN-2_1_6-en.msi
master@blingdenstone:~$ md5sum SimpLite-MSN-2_1_6-en.msi
875283db84f648da1d941c8a37835523  SimpLite-MSN-2_1_6-en.msi
master@blingdenstone:~$ sha1sum SimpLite-MSN-2_1_6-en.msi
6a92e676b89e7745ea1c6dbe89f3433be7d9daa2  SimpLite-MSN-2_1_6-en.msi
master@blingdenstone:~$
```

Confío en que a estas alturas comprendáis perfectamente el proceso (y podáis realizarlo desde Windows si ese es vuestro sistema): descargamos la clave de la desarrolladora (el enlace a la clave está en la página de descarga); importamos la clave a nuestro anillo local; comprobamos la firma (nos devuelve un error de confianza porque no hemos firmado esa clave) y después realizamos las comprobaciones habituales del tamaño del fichero y sus hashes. Ha llegado el momento de Instalarlo (obviamente para esto no quedará más remedio que ir a Windows... :-P).

Lo primero que nos encontramos es la típica pantalla de bienvenida al instalador seguida de un proceso de instalación de lo más normal, y que por tanto, obviaré detallar. Una vez finalizada la instalación, se inicia el asistente de configuración (Configuration Wizard), en el que se nos presentan las siguientes pantallas:

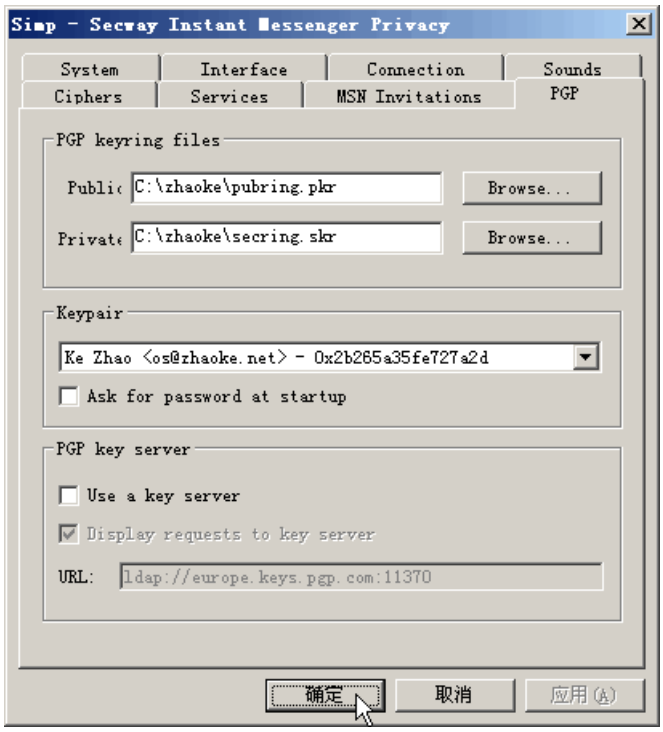
1) Pantalla de bienvenida. Nada nuevo.
2) Selección de "Ballon Boxes": la activamos.
3) Selección de tipo de conexión: en casi todos los casos (y si no sabéis qué elegir) seleccionaremos conexión directa (direct connection).
4) Aplicación cliente: Nos dejan elegir entre MSN Messenger, Trillian, Trillian Pro y otros (por si deseamos usarlo con otros clientes como Windows Messenger, aMSN, Gaim... aunque desconozco el grado de compatibilidad con los mismos). Seleccionaremos MSN Messenger.
5) Configuración automática: el software se configurará para quedarse listo para ser usado.

Tras este proceso, se iniciará el proceso de generación de claves.

¡Eh, un momento! ¿Pero esto no era OpenPGP? Pues sí y no. Me explico. El sistema de cifrado que utiliza SIMP Lite es muy, muy parecido al utilizado por OpenPGP: como podemos comprobar en las especificaciones del software (http://www.secway.fr/products/simplite_msn/tech.php?PARAM=us,ie), el sistema de cifrado se basa en claves RSA de hasta 2048 bits, así como cifrado simétrico de hasta 128 bits (AES o Twofish).

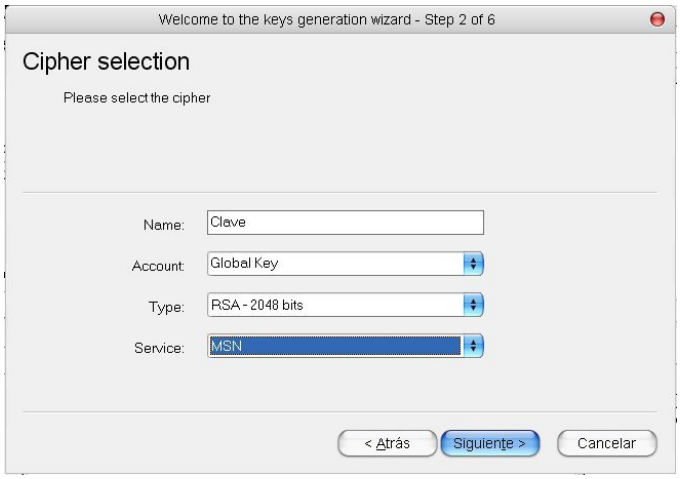
Aunque este sistema no es compatible -en principio- con OpenPGP, podría serlo perfectamente, y de hecho existe un plugin que implementa compatibilidad con PGP8... pero dicho plugin es para la versión Pro (de pago, 25$ la licencia individual) del software (http://www.secway.fr/products/simppro/spec/adv_plug.php?PARAM=us,ie). Echando un vistazo a las características de la versión pro (http://www.secway.fr/products/simppro/tech.php?PARAM=us,ie) encontramos las diferencias con la Lite.

Pero, si bien el sistema no es PGP, como ya he dicho, sí es prácticamente igual, por lo que la seguridad que nos otorga la versión gratuita es mucho más que suficiente... y más para la gente como yo que no usa Messenger. :-P

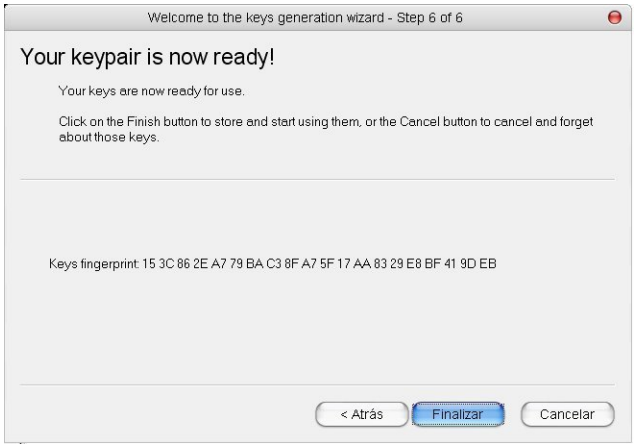


1) Asistente de generación de claves (Keys Generation Wizard).

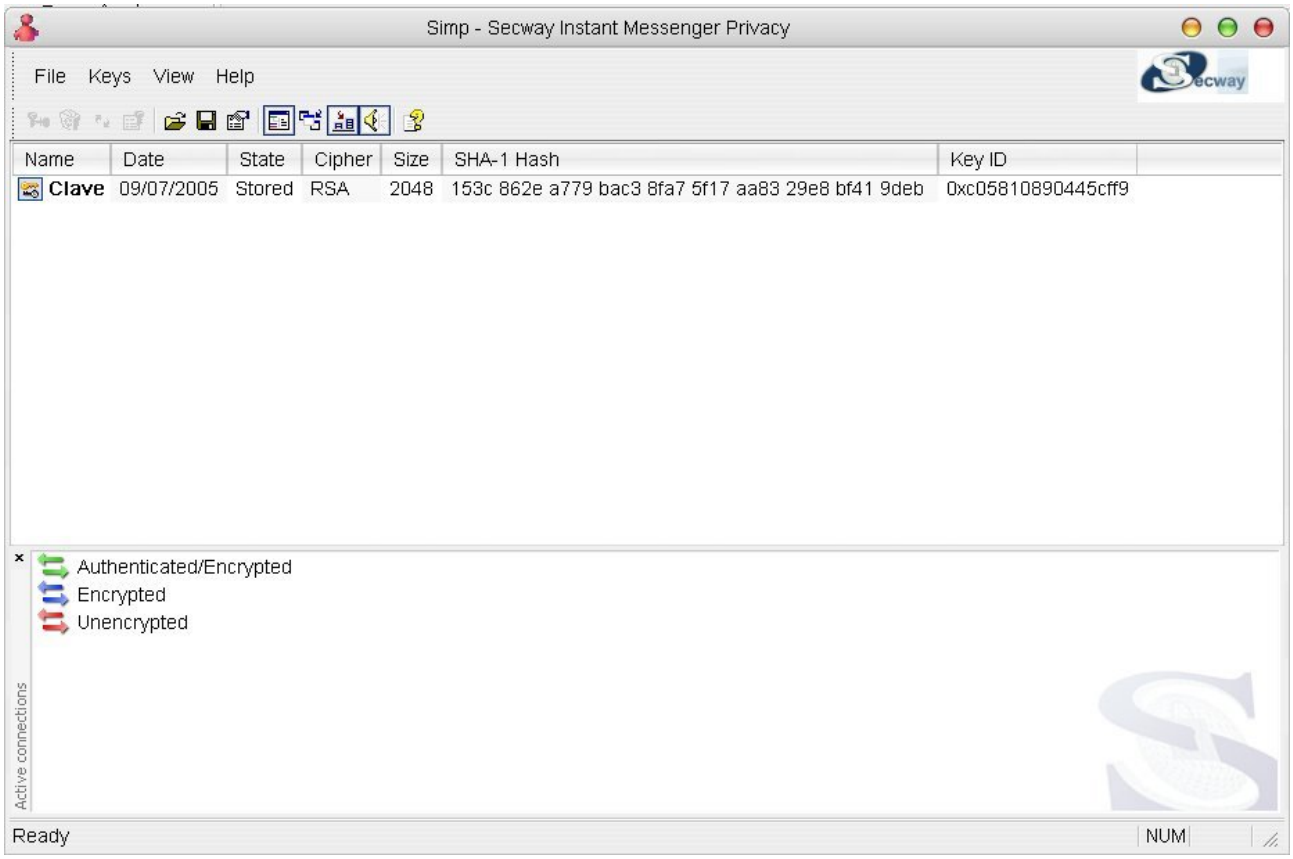
2) Selección del sistema de cifrado: debemos especificar el nombre de nuestra clave (yo, en un alarde de originalidad, la he llamado "Clave"), el tipo de cuenta (global), el tipo de cifrado (RSA de 2048 bits) y el tipo de servicio en el que vamos a utilizarla (MSN).



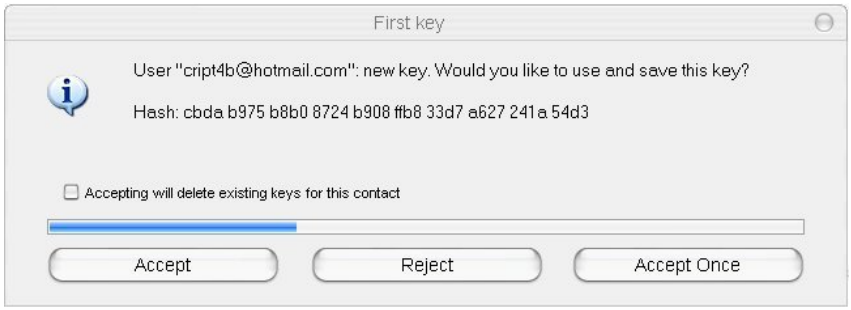
3) Pantalla de introducción de password: ¿no os recuerda una barbaridad a la de PGP? A mí sí jejeje...
4) Generación de entropía: como en otras ocasiones, la máquina necesita de entropía para poder generar números aleatorios para la generación de la clave.
5) Clave generada: ya tenemos nuestro par de claves y nos presenta el fingerprint (en forma de hash SHA-1) para comprobaciones.



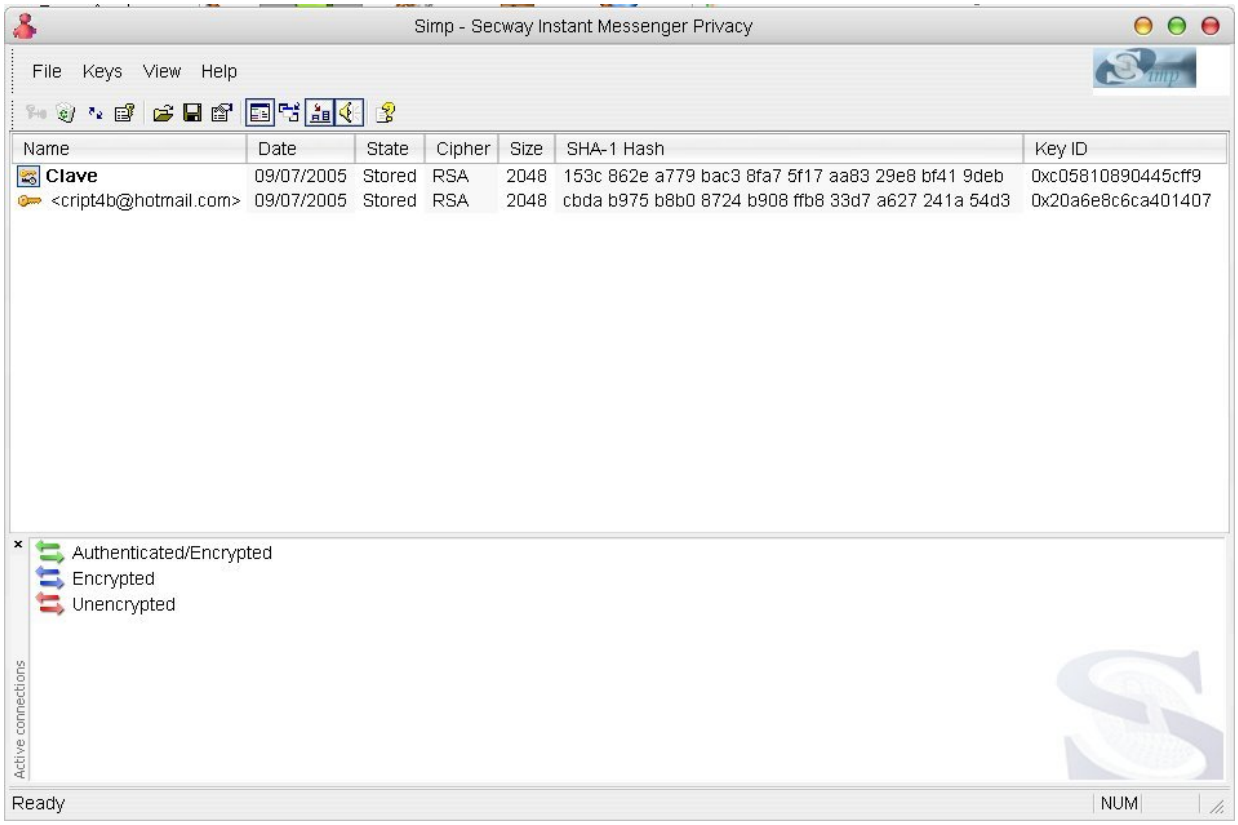
Es el momento de echar un vistazo a la pantalla principal de SIMP. Como veréis, tenemos un marco superior donde podemos ver las claves que tenemos en nuestro anillo, así como un marco inferior donde encontramos un diagrama de estado de las conexiones activas (cifradas y autenticadas, cifradas y no cifradas). SIMP se ha quedado a la escucha (podemos ver un icono en la bandeja de sistema) esperando a que iniciemos nuestro cliente de MSN Messenger...



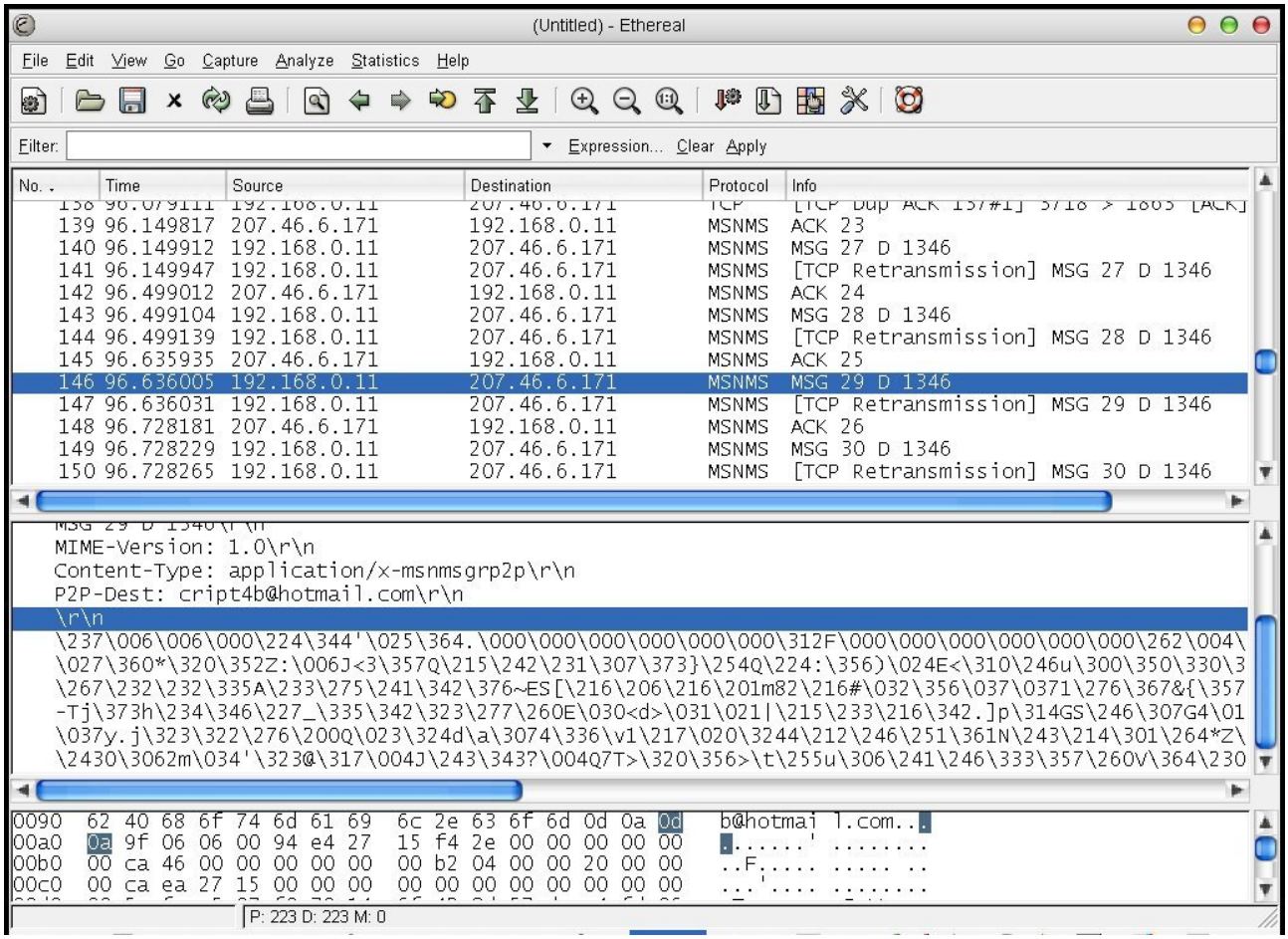
Nuestros queridos Usuario A y Usuario B, por consejo de su amigo Death Master, han instalado SIMP en sus máquinas para evitar fisgones en su red. Con sus claves generadas, han quedado a una hora concreta para tener una charla...



Usuario A inicia sesión y se encuentra con que Usuario B ya está conectado. Tanto mejor. Al abrir una ventana de conversación, aparece una ventana en la que se le solicita permiso para descargar la clave pública de su amigo Usuario B. Tras aceptar, Usuario A comprueba que la clave de su amigo ha sido incluida en su anillo local. Por fin, los dos amigos comienzan a charlar con absoluta tranquilidad, pues un mensaje emergente de SIMP les ha informado de que la conversación está cifrada y autenticada, lo cual han podido comprobar en el diagrama de conexiones activas de SIMP.



¿Y qué ha sido de nuestro atacante? Claro, él ha seguido su tónica habitual de cotillear en la red para espiar la conversación de nuestros dos amigos... pero ésta vez se ha encontrado con una sorpresa: la captura de datos es incomprensible. Revisando el tráfico capturado, se encuentra campos como "Session Key", lo cual es síntoma inequívoco de una conexión cifrada... como nuestro atacante es tremendamente malo, pero no tonto, se ha dado cuenta de que Usuario A y Usuario B, hartos de ser espiados, han puesto fin a la situación cifrando sus conversaciones. :-)



No sólo de MSN vive el hombre...

Efectivamente, MSN Messenger no es el único protocolo de mensajería instantánea, ni tampoco la mejor opción en mi opinión. Desde luego, si algún amigo mío está leyendo este artículo, aún debe estar alucinando por verme utilizar MSN Messenger (siempre me están dando la bronca porque no me conecto casi nunca...).

> *Efectivamente, no soy muy simpatizante de la mensajería por MSN Messenger. ¿Y porqué he hablado de él entonces? Porque aunque a mí no me guste, no puedo negar la realidad de que es el más utilizado por los usuarios... y este Taller que tienes en tus manos tiene un objetivo principal: proteger TU PRIVACIDAD.*
>
> *Por ello, he considerado importante tratar en primer lugar el cifrado de MSN con SIMP de forma detallada, y dejar para después lo que para mí habría sido la opción principal sin duda.*
>
> *Además, ten en cuenta que TODAS tus conversaciones de Messenger pasan por los servidores de Microsoft invariablemente... no sé a vosotros, pero eso a mí me pone los pelos como escarpias. :-P*

No voy a montar una comparativa de protocolos de mensajería instantánea, no es el momento ni el lugar, pero como ya sabréis, me gusta orientar mis artículos al mundo del software libre multiplataforma tanto como me es posible.

Por ello, vamos a hablar ahora de cifrado en la red Jabber con GnuPG. Si bien yo seguiré estos ejemplos desde mi sistema Debian SID GNU/Linux, pueden ser perfectamente seguidos desde cualquier otro sistema donde puedas compilar un cliente de Jabber con soporte OpenPGP y GnuPG. Y eso son muchos sistemas.

¿Qué es Jabber? Es un sistema de mensajería instantánea completamente libre (http://www.jabber.org/) cuyos protocolos basados en XML han dado lugar a una tecnología denominada XMPP -Extensible Messaging and Presence Protocol- (http://www.xmpp.org/). Este estándar ha sido descrito en los RFC's #3920 (http://www.ietf.org/rfc/rfc3920.txt) y #3921 (http://www.ietf.org/rfc/rfc3921.txt).

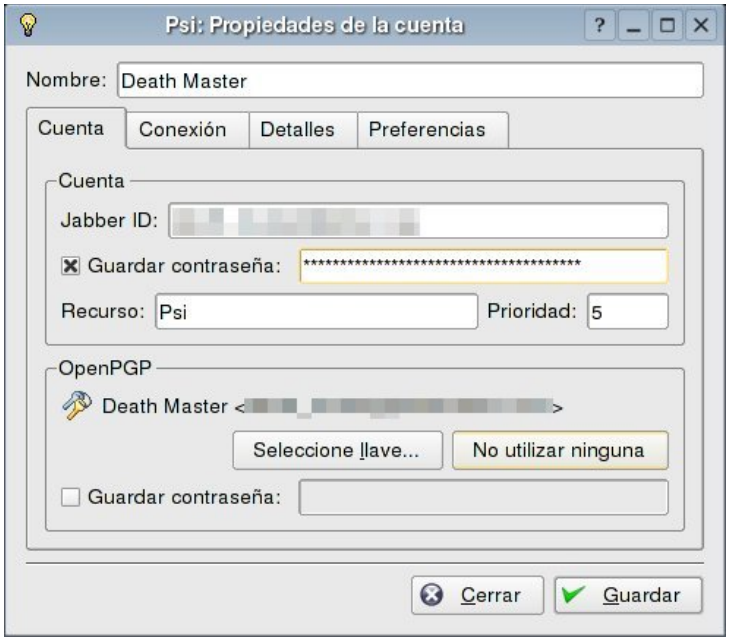
Las principales características de Jabber son: es un sistema abierto (libre), estándar (XMPP), descentralizado (no se "cae el servidor" porque hay muchos), seguro (aparte de OpenPGP, el propio XMPP implementa SASL y TLS), escalable (por el propio diseño de XMPP) y enormemente flexible.

Además, dispone de una enorme cantidad de clientes (http://www.jabber.org/software/clients.shtml) y servidores (http://www.jabber.org/software/servers.shtml).

No dejéis de probarlo. ;-)

De entre todos los clientes que soportan el protocolo Jabber, yo he seleccionado Psi (http://psi.affinix.com/) por ser mi favorito (entre otras cosas por tener soporte OpenPGP nativo). Las versiones de software que voy a utilizar son Psi 0.9.3-1 y GnuPG 1.4.1-1.

Al igual que con la mayoría del software libre, para obtener Psi podéis bajar binarios precompilados para vuestro sistema (para los Debianitas, apt-get install psi :-P) o bien compilar vosotros mismos el código fuente. Creo que nadie va a tener problemas con esto a estas alturas (y si los tiene... ¡visita nuestro foro y compártelos con nosotros!).

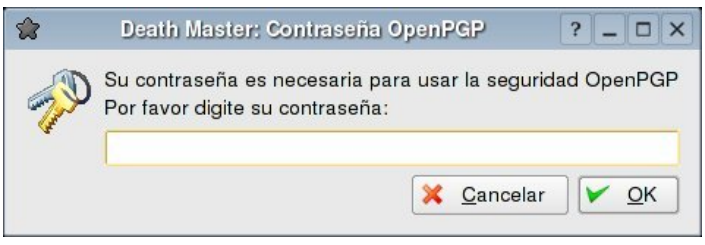


Una vez instalado Psi y creada una cuenta de Jabber (si no teníais ya una), seleccionamos la opción de "Modificar cuenta" y pinchamos en "Seleccionar llave". Psi nos mostrará una lista de las claves privadas disponibles en nuestro anillo de claves para que seleccionemos la que deseamos usar para Jabber. Recomiendo encarecidamente NO activar la opción de "Guardar contraseña" de OpenPGP... no olvidemos que estamos hablando de nuestra clave privada.

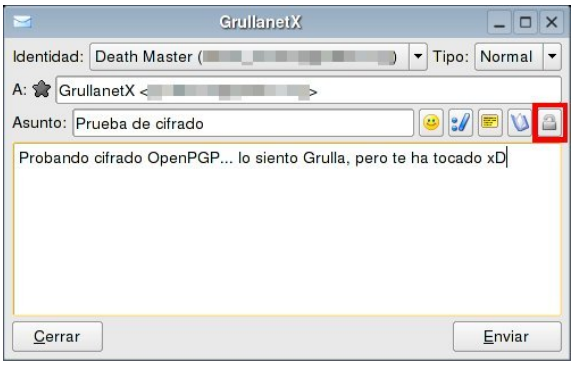
Para cualquier duda relacionada con el conjunto de protocolos de Jabber, con su funcionamiento, o cualquier asunto relacionado, os recomiendo encarecidamente que echéis un vistazo por JabberES (http://www.jabberes.org/), donde podréis encontrar tutoriales, guías, FAQ's...

Lamentablemente no disponemos de espacio para explicar paso a paso cosas como la generación de una nueva cuenta de Jabber... así que podéis buscar información en fuentes externas... o preguntar en nuestro foro. :-P

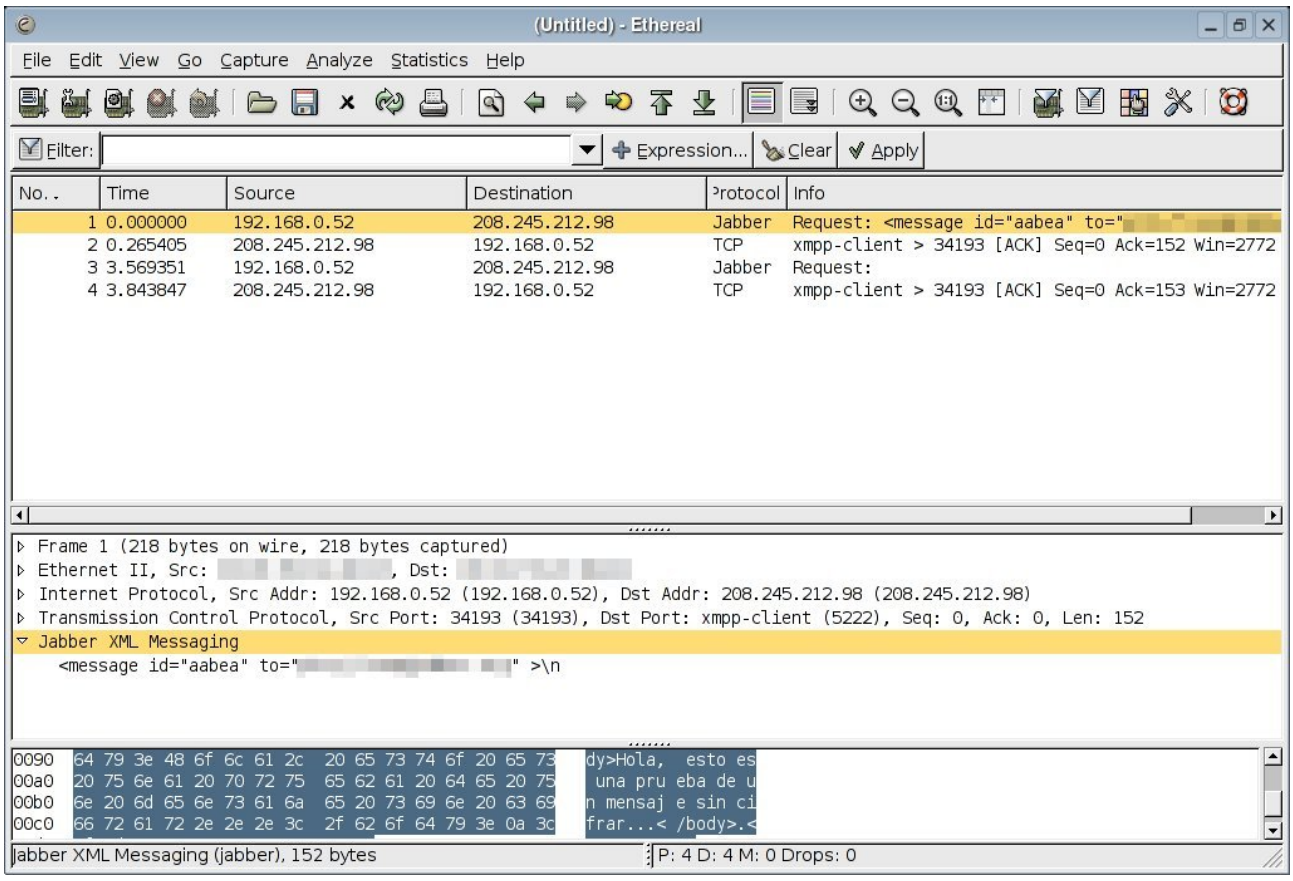
A partir de este momento, cada vez que iniciemos sesión, se nos solicitará el passphrase de la clave privada.

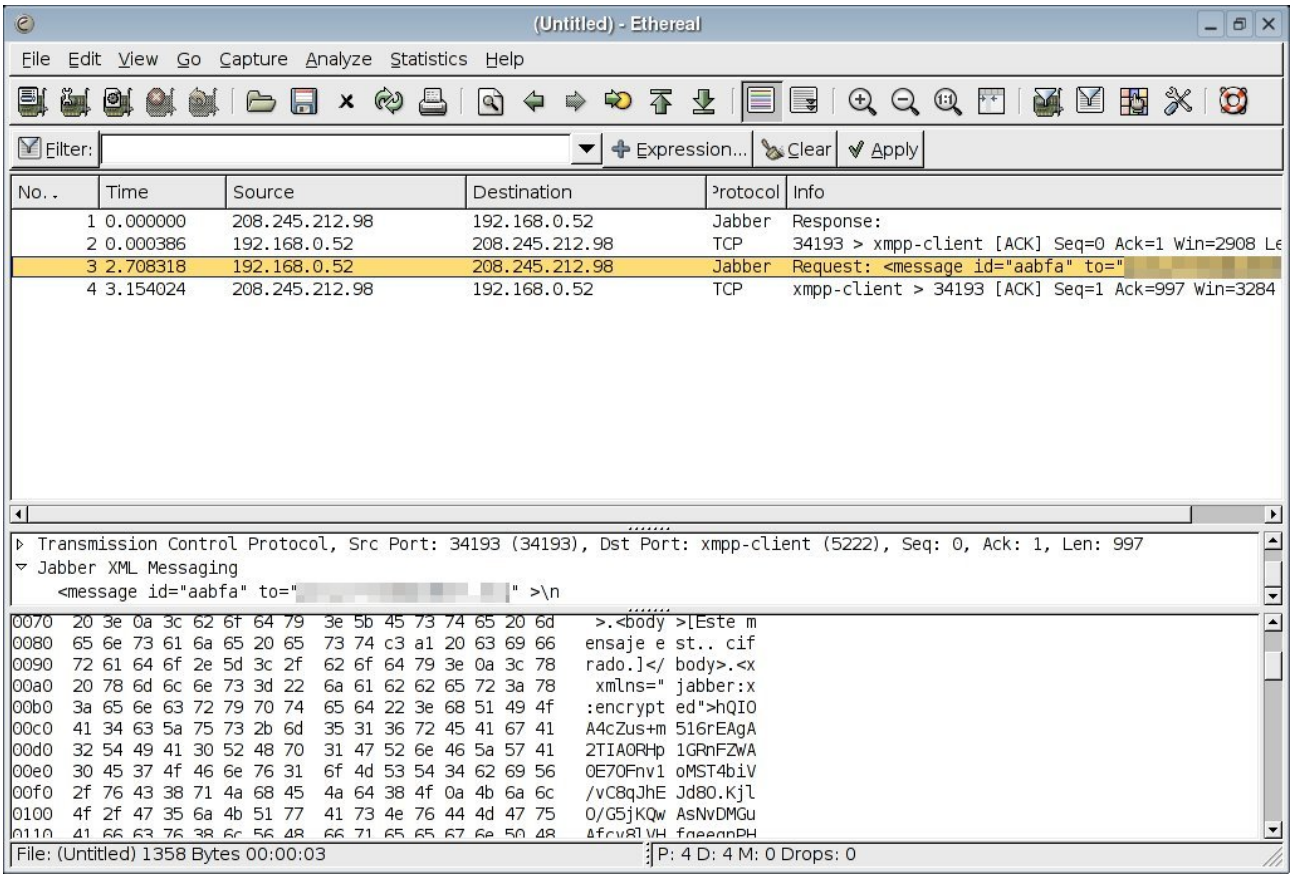


Tras asignar nuestra clave, debemos asignar las claves públicas a nuestros contactos. Para ello, abrimos el menú contextual del contacto y seleccionamos la opción "Asignar la llave OpenPGP". Ahora, cuando queramos enviar un mensaje o iniciar una charla con algún usuario de Jabber al que hayamos asignado una clave, sólo debemos pulsar en el icono de "Cambiar el cifrado" para activar el cifrado OpenPGP. Psi se encargará automáticamente de cifrar la información al destinatario y enviarla.

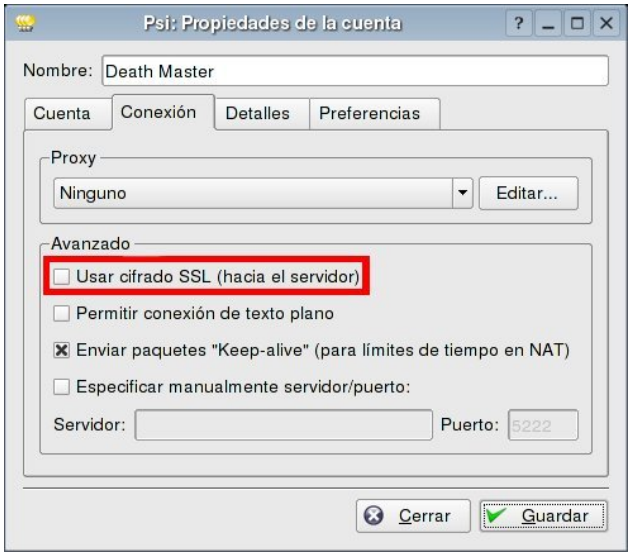


Como siempre, vamos a comprobar la efectividad de esta técnica. Cuando enviamos un mensaje sin cifrar, si capturamos el tráfico de la red, observamos que podemos leer completamente el mensaje y su destinatario, mientras que cuando enviamos el mensaje con la opción de cifrado OpenPGP en la captura observamos un mensaje cifrado... es decir, nada comprensible para un hipotético atacante. Como veréis, no he detallado mucho el tema del cifrado con Jabber porque creo que ya no es necesario, tras la experiencia (bastante similar) con el correo electrónico del artículo anterior y la de MSN Messenger, del cual hemos hablado hace unas páginas.





Sí quiero destacar que existe una opción adicional en Jabber que nos puede ayudar a incrementar (más aún) la seguridad de nuestras conexiones. ¿Para qué puede servir? Por ejemplo, en el caso de mandar mensajes cifrados con OpenPGP, al capturar el tráfico se podría ver quién es el destinatario del mensaje (al igual que ocurría con el correo electrónico). Podemos evitar esta incidencia usando conexiones cifradas (como SSL o TLS). Esta opción se encuentra disponible en la mayoría de los clientes de Jabber, como en Psi y de hecho en algunos como Pandion (http://www.pandion.be/) la comunicación con el servidor se realiza siempre mediante conexiones cifradas.



Y hasta aquí hemos llegado con este cuarto artículo del Taller de Criptografía. Espero, como siempre, que os haya resultado ameno pero sobre todo, que haya resultado útil... me gustaría pensar que desde aquí estoy contribuyendo a mejorar la privacidad de todos los internautas. :-)

¡Ah! Y una cosa para los usuarios de MSN Messenger: darle una oportunidad a Jabber, no os arrepentiréis.

Como siempre, si existen dudas o problemas con el artículo o algo relacionado podéis contactar conmigo o visitar el foro de la revista http://www.hackxcrack.com/phpBB2/index.php. Hasta el próximo artículo.

5- Blindando el disco duro

(Prevista su publicación en "PC Paso a Paso: Los cuadernos de Hack X Crack" número 30)
(No llegó a publicarse debido a la desaparición de dicha publicación)

Bienvenidos un número más al Taller de Criptografía. Durante las cuatro entregas anteriores del presente taller hemos centrado nuestros esfuerzos en aprender y -sobre todo- comprender el que es, hoy por hoy, el acercamiento más cómodo a la criptografía para el usuario medio: el sistema OpenPGP.

Algunas personas prefieren iniciar el estudio de la criptografía desde sus inicios, pero yo personalmente opino que si deseamos hacer de la criptografía una herramienta a nuestro servicio (como supongo que es el caso de la inmensa mayoría de vosotros) y no un fin en sí mismo (que, por otra parte, también es un objetivo muy loable, y que algunos compartimos) esto no tiene sentido: es más cómodo iniciarnos en el uso de sistemas ampliamente usados, extensivamente probados, y suficientemente sencillos como para comprenderlos sin una fuerte base teórica de criptología (campo que engloba tanto a la criptografía como al criptoanálisis, del que nada hemos comentado ni comentaremos al menos de momento).

Así pues, ya conocemos OpenPGP bastante bien, o al menos lo suficiente... pero la criptografía informática no empieza ni acaba en OpenPGP, aunque sea una parte importante de la misma. Por todo ello, a partir de ahora vamos a "independizarnos" (criptográficamente hablando, de lo otro al precio que están los pisos... :-P) de OpenPGP, y a utilizarlo únicamente cuando sea necesario para nuestros distintos objetivos.

Así es, porque a partir de ahora trataremos diversos aspectos de la seguridad informática que, de una u otra forma, precisan de los servicios de la criptografía para su correcto funcionamiento. Esos serán nuestros objetivos.

Y nuestro objetivo de hoy es blindar nuestro disco duro.

La noche de los discos duros vivientes

Durante un período de dos años comprendido entre 2000 y 2002, dos estudiantes del MIT (http://web.mit.edu/) realizaron un estudio sobre discos duros usados adquiridos mediante tiendas de subastas en Internet o tiendas de segunda mano.

De los resultados desprendidos del citado estudio se llegó a la conclusión que, al igual que los muertos de la película de George A. Romero, los datos que creíamos muertos en nuestros discos duros pueden levantarse y decir muchas cosas que no quisiéramos que contaran...

Para el estudio adquirieron 158 discos duros, de los cuales 129 eran operativos. Primera moraleja: al comprar componentes de segunda mano hay que estar atento, pues en este caso había casi un 20% de discos estropeados. Hagamos un desglose de lo que ocurrió con esos 129 discos:

- 83 discos (64%) contenían particiones FAT (FAT16 ó FAT32) accesibles. De ellos 51 (40% del total) habían sido formateados. De todos ellos se pudo obtener información.

- 46 discos (36%) no contenían particiones accesibles. De 30 (23%) se pudo obtener información.

- 12 discos (9%) habían pasado por procesos de limpieza seguros. De ninguno de ellos se pudo obtener información.

Si echamos las cuentas, veremos que en un 87% de los casos se pudo obtener información de un disco duro del que su dueño se había deshecho. No sé a vosotros, pero a mí me parece una auténtica barbaridad.

¿Qué tipo de información fue recuperada? 675 ficheros DOC, 274 XLS, 20 PST, 566 PPT... gran parte de la información carecía de importancia: archivos normales, ficheros de la caché de navegadores web, pornografía...

Pero había mucha otra información que tenía importancia, y mucha: datos médicos de un hospital para niños, datos de empleados de varias empresas... y más de 6000 números de tarjetas de crédito, de los cuales 2800 pertenecían a un disco que más tarde se comprobó que había pertenecido a un cajero automático. Más barbaridades.

Volvemos a un tema del que ya he hablado en algunas ocasiones... la importancia de la seguridad es relativa: a mí me da igual borrar ciertos documentos (PDF, películas, música...) y que puedan ser eventualmente recuperados, pero desde luego hay ciertos ficheros personales más sensibles que no me gustaría que me robaran (contraseñas, datos bancarios, bases de datos...).

Pero es que este tema no atañe únicamente a usuarios privados: estamos hablando del disco duro de un cajero automático que -vaya usted a saber cómo- ha llegado al mercado de segunda mano y que contiene gran cantidad de números de tarjetas de crédito. Seguramente muchos de vosotros estaréis aún alucinando con esta información, y sin duda todos estaréis empezando a tomar conciencia de la importancia de proteger los datos privados.

Vida más allá de la papelera de reciclaje

Pero, ¿cómo es eso de recuperar datos de un disco duro? Coged a un usuario medio de PC y preguntarle qué haría para recuperar un dato borrado accidentalmente. Irá como una flecha a la papelera de reciclaje y lo buscará. Y si no está, no hay nada que hacer, porque el fichero ha atravesado ya la singularidad del agujero negro de la papelera y está en la censura cósmica particular de nuestro disco duro. Pues va a ser que no.

Para entender porqué no (porque a nosotros no nos interesa tanto el qué como el porqué, ¿verdad? ;-P) hay primero que comprender cómo funciona un disco duro. Aunque hace ya mucho de los primeros discos duros de unos pocos Mb, allá por los tiempos de los 286 (cuando los dinosaurios poblaban Silicon Valley), su tecnología es básicamente la misma. De hecho, su base es muy similar a las enormes cintas que usaban los primeros ordenadores, o de las -ya jubiladas- cintas de cassette de música. Todo se lo debemos al electromagnetismo.

Básicamente un disco duro se compone de una serie de discos recubiertos de una fina capa de algún tipo de aleación metálica (típicamente óxido de hierro o de cobalto) que giran entorno a un eje, y una serie de cabezas lectoras/escritoras. Estas cabezas son las encargadas de escribir (magnetizando el material de la superficie del disco) y de leer (detectando la magnetización de dicha superficie) los datos.

> *Esta explicación puede ayudar a comprender grosso modo cómo funciona un disco duro, pero ni mucho menos es rigurosa. Si alguien está interesado en aprender más sobre este importante dispositivo, os recomiendo hacer una visita a nuestro bien amado google...*

A diferencia de las antiguas cintas de acceso puramente secuencial, los discos duros tienen un sistema de acceso directo (para que los puristas no me cuelguen, diré que en realidad no es acceso directo, pero a efectos prácticos se le parece bastante). Esto significa que los datos no tienen porqué ser almacenados uno detrás de otro, como efectivamente ocurre. La forma en que se manejan los datos depende completamente del sistema de ficheros utilizado.

> *Sobre los sistemas de ficheros hay MUCHO que contar. Cómo se escriben, se borran o se buscan los datos en la partición depende totalmente del sistema de ficheros utilizado en la misma. Es más, como más adelante veremos, las posibilidades de recuperación de datos borrados dependen también de ello.*
>
> *Los sistemas de ficheros más comunes son: FAT16 (MS-DOS), FAT32 (Windows 95 OSR2), NTFS (Windows NT), ext3 (Unix), ReiserFS (Unix).*
>
> *Si usáis Windows XP ó Windows 2000 es altamente probable que vuestro sistema de ficheros sea NTFS, mientras que versiones anteriores de Windows utilizan FAT32 (algunos usamos aún FAT32 con XP...). Si utilizáis algún Unix (como GNU/Linux o xBSD) lo más probable es que uséis ext2, ext3 o ReiserFS.*

> *Si alguien está interesado en este tema, recomiendo encarecidamente visitar este enlace de la Wikipedia y bucear por los distintos enlaces que contiene. Tenéis para estar entretenidos unas cuantas horas.*
>
> *http://en.wikipedia.org/wiki/Comparison_of_file_systems*

Aunque cada sistema de ficheros es "de su padre y de su madre" (o de su diseñador/diseñadora en este caso), todos tienen algo en común: un índice de los datos.

Imaginad qué hacéis cuando deseáis consultar en un libro (una novela no sirve, se supone que se lee secuencialmente de principio a fin. Cogemos el libro "Estudio de los embudos Wadalbertitas" para buscar información sobre el Efecto Funel. Lo primero que hacemos es buscar en el índice y dirigirnos a la página en cuestión para leer lo que buscábamos.

Los discos duros funcionan igual, existe un índice del contenido del mismo que se actualiza dinámicamente para saber qué hay en el disco y dónde está. El cómo se llama este índice y cómo se organiza depende del sistema de ficheros: en FAT es una lista enlazada para ficheros y una tabla para la estructura de los directorios; en NTFS y ReiserFS es un árbol B+ (un tipo de árbol binario) tanto para ficheros como directorios...

> *¿Por qué es necesario dicho índice? Porque un disco duro se asemeja a un libro en el que escribimos donde nos viene en gana. O mejor dicho, donde hay hueco. Y si deseamos escribir treinta páginas y encontramos diez libres, las escribimos y las demás se ponen por otro lado donde haya sitio.*
>
> *Esto es lo que da lugar al fenómeno de la fragmentación del disco (¿os suena el desfragmentador de Windows?), aunque también depende de cómo se organice el sistema de ficheros. En ext3, por ejemplo, no es necesario el desfragmentado de disco.*

Ahora pensemos en el significado de una situación práctica habitual. Imaginad que estáis copiando unas películas de un DVD al disco duro (¡Eh, tú! Te he visto, ahora no minimices el eMule y mires para otro lado disimulando que nos conocemos...). Lo normal es que el ordenador se tome su tiempo en copiar todos los datos al disco duro. Después de hacer lo que sea con todos esos datos, los borramos... y ¿qué pasa? Lo que tardó en copiarse 5 ó 10 minutos se ha borrado en segundos. Lo mismo pasa cuando instalamos/desinstalamos software pesado.

¿Qué ocurre para que eliminar datos sea tan rápido frente a escribirlos? Teóricamente el proceso sería similar: al escribir buscaríamos en el índice, iríamos a los sectores libres, escribiríamos la información, actualizaríamos el índice... y al borrar debería ser el proceso inverso. Pues no.

La mayoría de los sistemas de ficheros, cuando borramos datos, únicamente eliminan la información de donde están dichos datos del índice. Pero los datos están intactos. El sistema operativo considera ese espacio vacío, y cuando necesite escribir algo, esos sectores serán usados para ello... pero mientras tanto los datos siguen ahí. Es decir, hay vida más allá de la papelera de reciclaje.

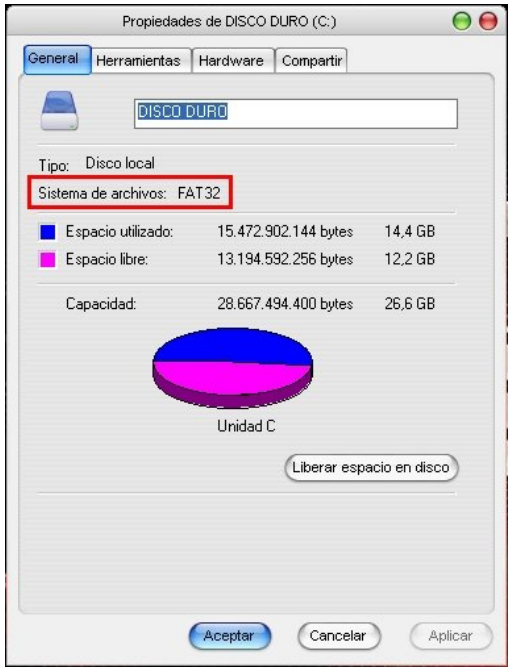
Ahora vamos a ver cómo echarle el guante a esos datos...

Recuperando los datos del limbo

La mejor forma de asimilar algo es practicándolo (y es que, a andar se aprende andando...), eso es lo que vamos a hacer.

Esta pequeña práctica vamos a realizarla sobre un sistema Windows (en mi caso particular, Windows XP con Service Pack 1, pero cualquier versión de Windows sirve) en una partición FAT32. El motivo de elegir estas condiciones es principalmente porque es donde mejor vamos a apreciar los resultados, dado que en un sistema Linux bajo ext3, la recuperación de datos es mucho más complicada debido al uso que este sistema de ficheros realiza de los i-nodos.

¿Cómo sé cuál es mi sistema de ficheros en Windows? Muy fácil, sólo debes acceder a las propiedades de la partición que desees observar (desde Mi PC o el explorador de Windows). En la parte superior aparecerá un campo llamado "Sistema de archivos" donde podréis comprobar el vuestro.



Lo primero que vamos a necesitar es un programa gratuito (freeware, no confundir con software libre porque no es lo mismo :-P) llamado PC Inspector File Recovery. Podemos obtener dicho programa de su página oficial: http://www.pcinspector.de/file_recovery/es/welcome.htm . Pinchando en la pestaña "Descarga" accedemos a dicha sección, donde podremos descargar la versión 4.0 de este software. Como siempre, lo primero es comprobar el tamaño y los hashes para saber que tenemos el mismo fichero.

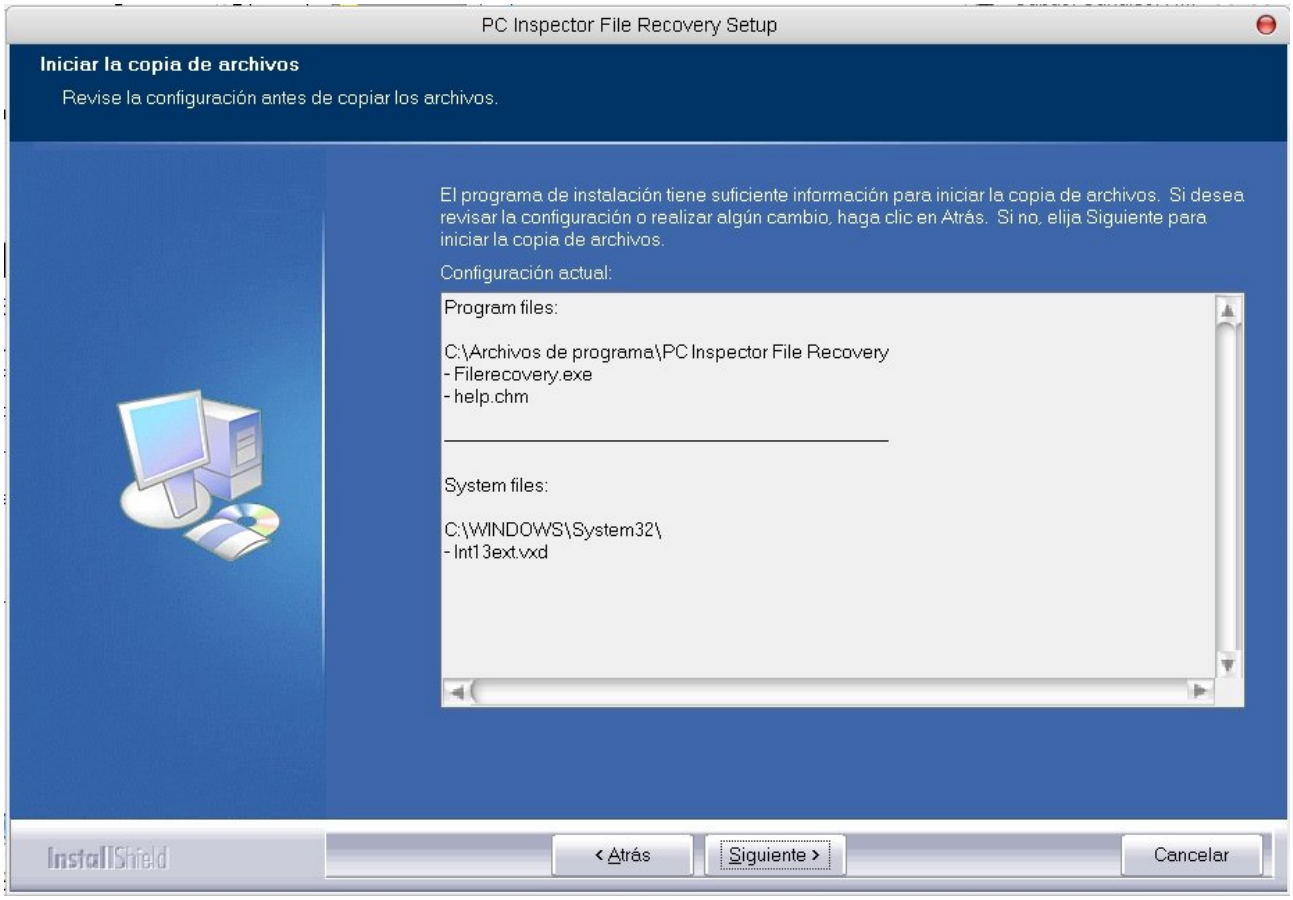
```
C:\WINDOWS\system32>dir "C:\Mis Documentos\pci_filerecovery.exe"
{...}

17/08/2005  17:14         6.113.439 pci_filerecovery.exe
{...}
```

```
C:\WINDOWS\system32>md5 "C:\Mis Documentos\pci_filerecovery.exe"
523124817377308B91180FB5CD50AD94
C:\Mis Documentos\pci_filerecovery.exe

C:\WINDOWS\system32>sha1 "C:\Mis Documentos\pci_filerecovery.exe"
NAME: C:\Mis Documentos\pci_filerecovery.exe HASH:
87fd9edf8f3ce2de7cffc4edbd0a954def0fbec0

C:\WINDOWS\system32>
```

Recordemos que en la entrega anterior del taller trabajamos con un programa con interfaz gráfica para calcular cómodamente los hashes de un fichero. El programa es HashCalc, y puede ser descargado desde: http://www.slavasoft.com/hashcalc/.

Con él podréis comprobar dichos hashes sin necesidad de usar la línea de comandos ni descargar md5.exe o sha1.exe (que no vienen incluidos con Windows).

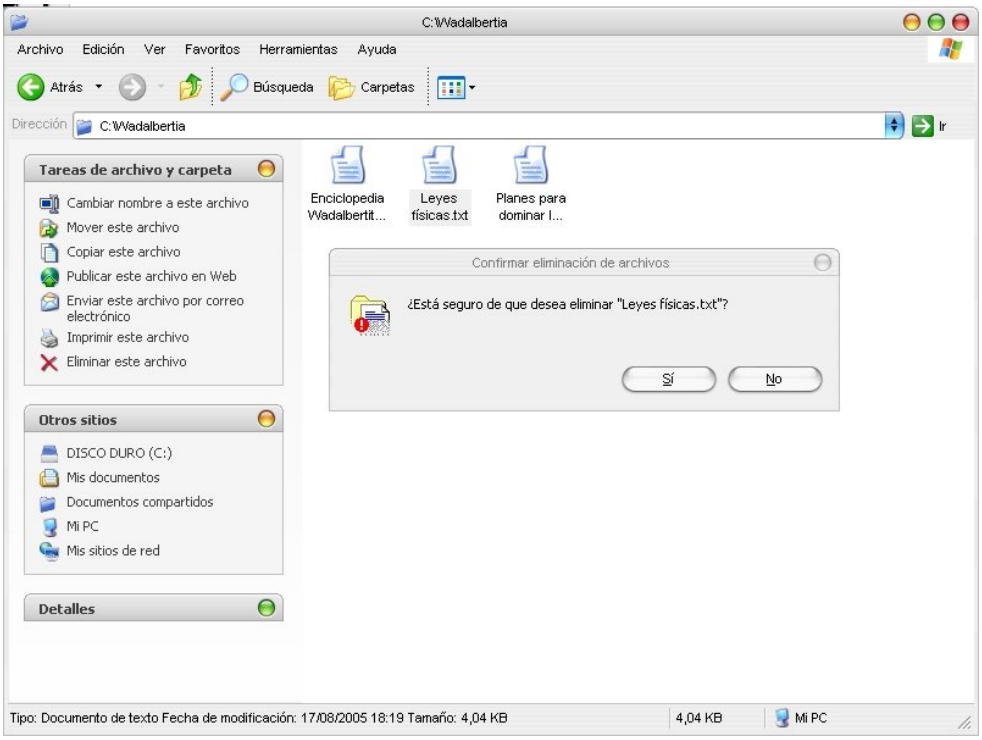
La instalación del software no debería comportar ningún problema. Es el momento de ponernos manos a la obra.



Echemos un vistazo al contenido de la carpeta donde realizaremos las prácticas.

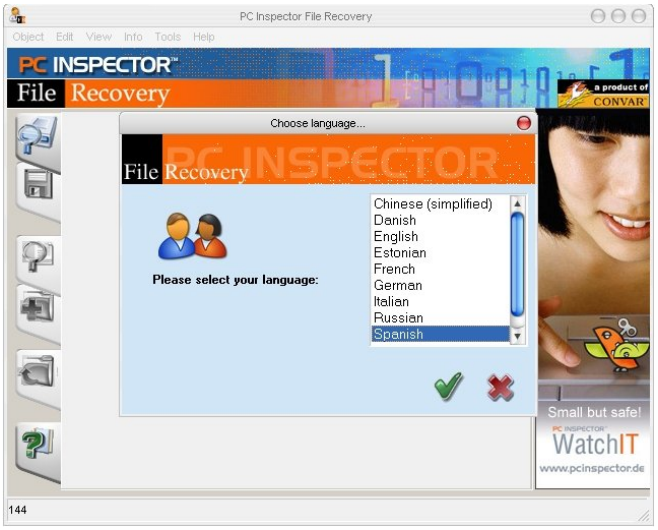
```
C:\Wadalbertia>dir
 El volumen de la unidad C es DISCO DURO
 El número de serie del volumen es: B043-3C33

 Directorio de C:\Wadalbertia

17/08/2005  18:12    <DIR>          .
17/08/2005  18:12    <DIR>          ..
17/08/2005  18:14            36.059 Enciclopedia Wadalbertita.txt
17/08/2005  18:18               308 Planes para dominar la Intesné.txt
17/08/2005  18:19             4.140 Leyes físicas.txt
               3 archivos         40.507 bytes
               2 dirs  13.190.496.256 bytes libres

C:\Wadalbertia>
```

Decidimos que las leyes físicas de Wadalbertia es algo demasiado secreto y el fichero “Leyes físicas.txt” debe ser eliminado. Por ello, vamos a borrar el fichero... no, nada de enviarlo a la papelera de reciclaje, sino eliminarlo completamente.



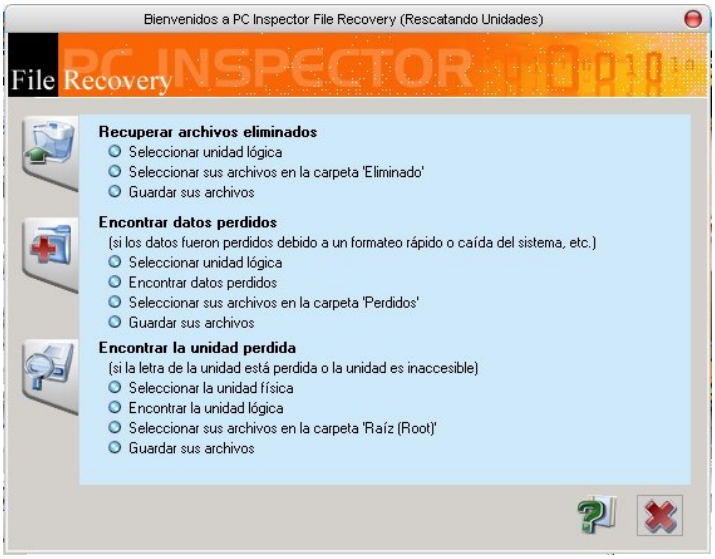

```
C:\Wadalbertia>dir
 El volumen de la unidad C es DISCO DURO
 El número de serie del volumen es: B043-3C33

 Directorio de C:\Wadalbertia

17/08/2005  18:12    <DIR>          .
17/08/2005  18:12    <DIR>          ..
17/08/2005  18:14            36.059 Enciclopedia Wadalbertita.txt
17/08/2005  18:18               308 Planes para dominar la Intesné.txt
               2 archivos         36.367 bytes
               2 dirs  13.189.251.072 bytes libres

C:\Wadalbertia>
```

Para borrar directamente un fichero, solamente hay que pulsar la tecla de mayúsculas (shift) mientras seleccionamos la opción eliminar del menú contextual del fichero (que aparece al pulsar con el botón derecho del ratón sobre él) o bien pulsamos la tecla de borrado (supr).



Puede que creamos que ahora esos datos están a salvo... pero nada más lejos de la realidad. Es el momento de que entre en acción PC Inspector File Recovery.

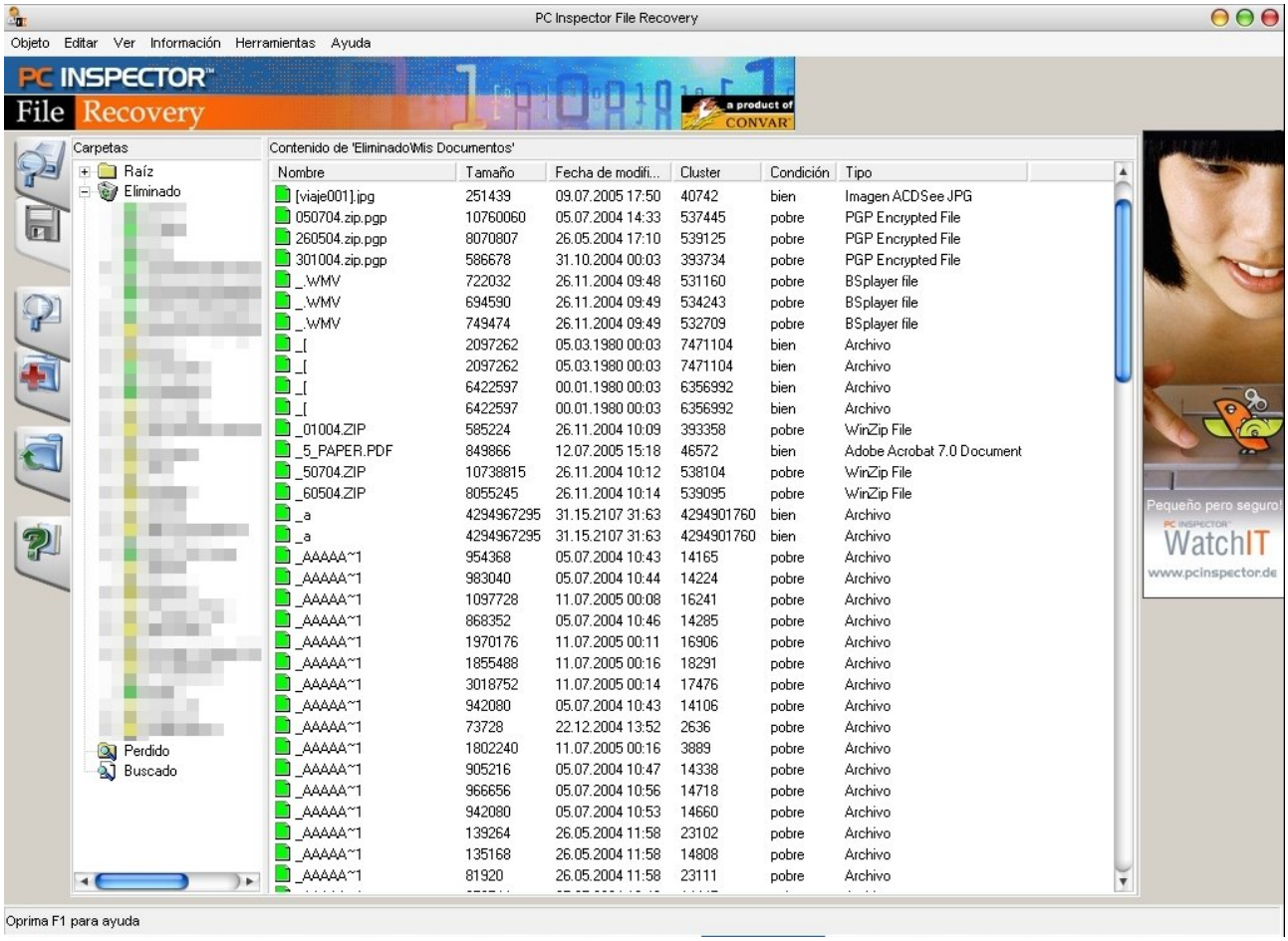
Nada más abrir el programa, nos pedirá que seleccionemos el idioma en el que deseamos ejecutar la aplicación. Tras seleccionar castellano (o el que queráis, yo seguiré el ejercicio en castellano), se nos presenta ante nosotros un menú de selección con las tres opciones principales del programa, así como una breve descripción del proceso a seguir con cada una de ellas.



La opción “Recuperar archivos eliminados” nos permite rescatar un archivo eliminado intencionadamente, y es la opción que vamos a utilizar en nuestro ejemplo. “Encontrar datos perdidos” permite buscar archivos que han sido eliminados por un fallo del software o el sistema operativo. “Encontrar unidad perdida” sirve para recuperar particiones completas tras un fallo en la tabla de particiones.

Seleccionamos la primera opción pulsando en el icono correspondiente para encontrarnos un diálogo en el que debemos seleccionar una unidad lógica (partición) en la que trabajar. El programa realiza un escaneo automático de las unidades físicas y realiza una lista de las unidades lógicas. Es importante reseñar que este software trabaja con sistemas de ficheros FAT o NTFS. Además de la partición FAT32, el disco duro que estoy utilizando (el de mi portátil) tiene otras dos particiones: una con sistema ext3 y otra con Linux swap.

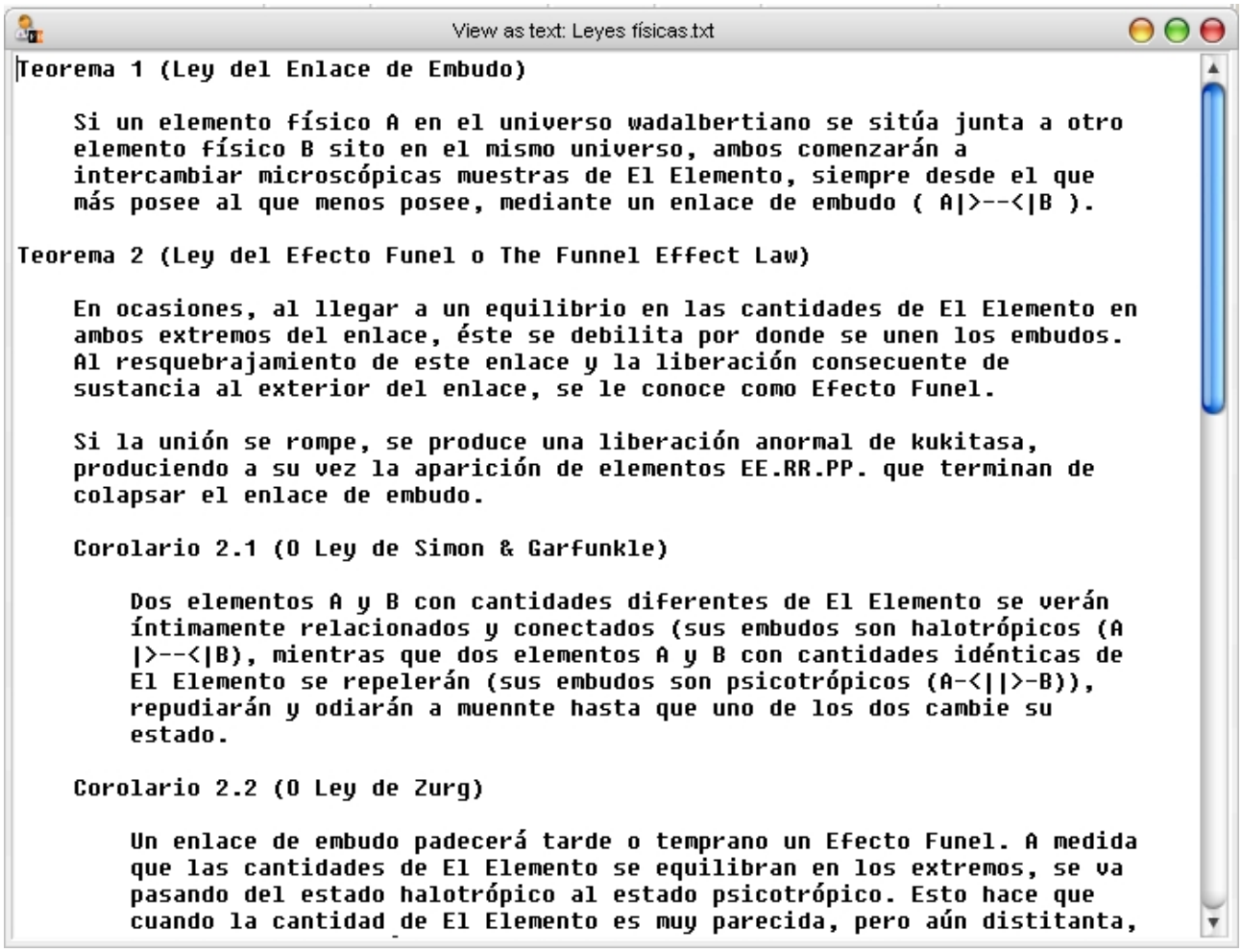
Una vez seleccionada la unidad, pinchamos en el icono verde de aceptar y se carga un árbol de directorios en el que podemos ver ciertas carpetas en color normal (que existen en nuestro sistema actualmente) y otras en color verde (que han sido eliminadas de nuestro sistema). Dentro de ellas puede que veamos algunos ficheros en color verde, que representarán los ficheros eliminados. En esa lista tenemos información como el nombre del fichero (o lo que quede del nombre), el tamaño, fecha de modificación, cluster donde ha sido encontrado, condición del fichero y tipo.



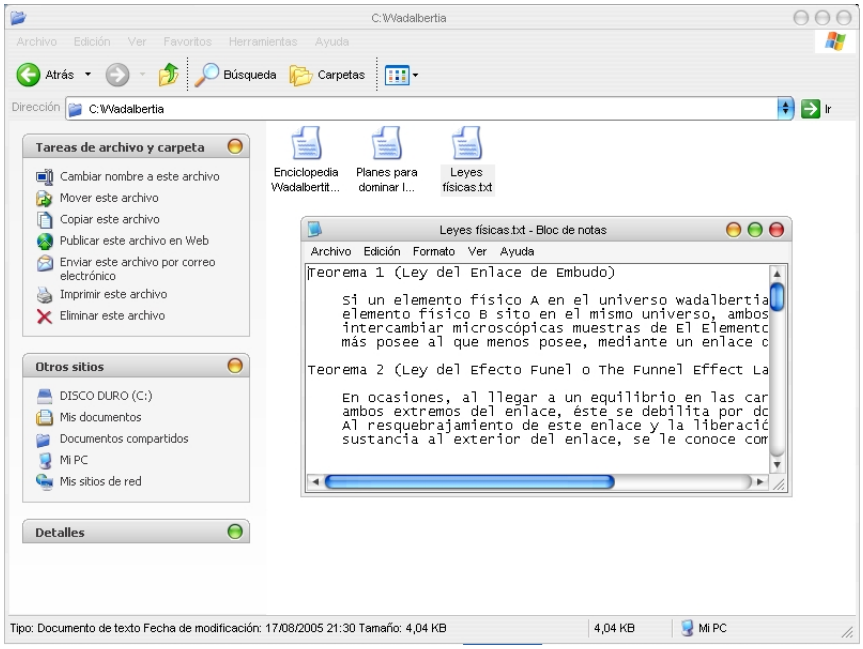
Si el fichero tiene buen estado, es muy posible que podamos recuperar bastante información de él. Si por el contrario tiene un estado pobre, será complicado sacar nada en claro de él.

Si en lugar de recuperar archivos eliminados, usamos el programa para encontrar datos perdidos, no podremos saber el estado de los ficheros encontrados hasta que los abramos.

Nos dirigimos a la carpeta donde estamos realizando las pruebas y nos encontramos con el fichero que acabamos de eliminar. Pulsamos sobre él con el botón derecho y seleccionamos “verlo como texto”. Nos encontraremos ante un diálogo en el que deberemos seleccionar cómo leer los clusters Seleccionamos noFAT y aceptamos. Ante nosotros tenemos el fichero intacto.



Si no habéis logrado recuperar el fichero no os preocupéis: es normal. Dado que he utilizado un fichero muy pequeño para este ejemplo (4 Kb), es muy fácil que cualquier escritura en el disco -por ejemplo, un archivo temporal- sobrescriba la información y la estropee.



Si seleccionamos ahora la opción “guardar en”, podremos recuperar el fichero a nuestro disco duro.

Recuperar información de ficheros pequeños es relativamente complicado, pero recuperar PARTE de la información de un fichero de gran tamaño es algo muy sencillo. En mi disco duro he encontrado ficheros en buen estado que corresponden a películas que vi y eliminé hace un par de meses, lo cual nos puede dar una idea de la precariedad de la seguridad de los datos...

Resulta irónico pensar en la cantidad de oficinas que cuentan con desfibradora de papel para eliminar documentos de forma segura, pero que cuando deciden renovar el parque de PC's, venden los ordenadores viejos, o permiten a los empleados quedárselos... o los tiran a la basura y se quedan tan tranquilos.

En cualquier caso, PC Inspector File Recovery no es el único software de recuperación de datos ni muchísimo menos... hay una gran cantidad de programas orientados a este fin: Fundelete (http://www.sysinternals.com/Utilities/Fundelete.html), GetDataBack (http://www.runtime.org/gdb.htm), Zero Assumption Recovery (http://www.z-a-recovery.com/), Tune Up Utilities 2004 (http://www.tune-up.com/products/tuneup-utilities/), Steganos Security Suite (http://www.steganos.com/), File Scavenger (http://www.quetek.com/prod02.htm), R-Linux (http://www.data-recovery-software.net/Linux_Recovery.shtml), etc. Como siempre, con buscar un poco por la red encontraréis decenas de programas.

Borrado seguro de datos

¿Qué podemos hacer, pues, para librarnos definitivamente de los datos de nuestro disco duro? La única manera de eliminar de verdad esos datos es sobrescribiéndolos con nuevos datos. ¿Significa esto que simplemente con editar el fichero y sobrescribir su información sirve? No.

Imaginad que tenemos un fichero de texto de 1 Mb que queremos borrar definitivamente, para lo cual lo editamos y escribimos 1 Kb de texto. ¿Qué creéis que ha pasado con los 1023 Kb restantes? Si estáis pensando que aún están ahí... estáis en lo cierto. Incluso puede que al sistema operativo le haya salido más "barato" no mover los cabezales del disco y escribir ese kilobyte en algún sitio aparte y estén los 1024 Kb intactos.

Vale, sobrescribir "a mano" no sirve. ¿Desfragmentar? Tampoco. Quizá, como mucho, sirva para librarse de ficheros pequeños en un disco altamente fragmentado, pero poco más.

> *Para entender esto último un poco mejor veamos un ejemplo teórico. Imaginemos cinco ficheros en un disco.*
>
> [A][A][B][B][B][B][C][C][C][D][E][E][E][E]
>
> *Borramos los ficheros A, D y E.*
>
> [a][a][B][B][B][B][C][C][C][d][e][e][e][e]
>
> *Ahora escribimos un fichero F que ocupa cuatro unidades de espacio.*
>
> [F][F][B][B][B][B][C][C][C][F][F][e][e][e]
>
> *Y desfragmentamos el disco.*
>
> [F][F][F][F][B][B][B][B][C][C][C][e][e][e]
>
> *Como vemos, los ficheros A y D desaparecen completamente, pero el fichero E está aún parcialmente en memoria.*

Desfragmentar tampoco... ¿formatear el disco? Depende... un formateo rápido obviamente no. Con cualquier herramienta de recuperación de datos podremos obtener muchísima información. Un formateo estándar de alto nivel tampoco servirá. Y un formateo de bajo nivel (reconstruir completamente la estructura física del disco) tampoco.

¿¿Y si ponemos cada bit del disco duro a 0?? Pues tampoco. Estaréis pensando que estoy chalado, que la falta de sueño y el exceso de café me empieza a pasar factura... puede, pero eso no cambia que esa opción tampoco sirva. Además, después del ataque Phreaking van Eck, también conocido como TEMPEST(http://en.wikipedia.org/wiki/Van_Eck_Phreaking), que comentamos en la primera entrega del presente taller, deberíais estar ya curados de espanto... ;-)

Como hemos comentado al principio, los discos duros basan su funcionamiento en el electromagnetismo. La cabeza lectora/escritora magnetiza una traza de metal para codificar una información, que después puede ser leída e interpretada atendiendo a dicha magnetización. Si habéis estudiado el electromagnetismo, sabréis que dos campos diferentes a los que se les aplica una misma carga dan como resultado dos campos diferentes. ¿Qué quiere decir esto? Que dos bits del disco duro diferentes, cuando son magnetizados para representar una misma información, tendrán campos magnéticos ligeramente diferentes. La diferencia es infinitesimal, pero sólo sería necesario el equipo lo suficientemente sensible y las suficientes ganas (y medios) para llevarlo a cabo.

Entonces, ¿no hay nada que pueda hacerse para borrar definitivamente una información? Siempre hay algo que pueda hacerse. :-)

La solución es muy sencilla: dado que, como hemos comentado, dos campos distintos a los que se aplica una misma carga nos devuelven dos campos diferentes... apliquemos cargas diferentes. Es decir, que sobrescribiendo los sectores que almacenan los datos a borrar con información aleatoria unas cuantas veces, será absolutamente imposible recuperar dichos datos. Hasta aquí perfecto, ahora viene la parte complicada.

Para nosotros obtener datos aleatorios es muy sencillo: lanzando dados o monedas. Las leyes físicas harán el resto para crear un sistema caótico donde no puedan predecirse los resultados. Pero un ordenador, al igual que Dios según Einstein, tampoco juega a los dados. Es más, el ordenador no puede hacerlo.

Dado que los ordenadores actuales, en contraposición a los computadores cuánticos, son máquinas deterministas que ejecutan algoritmos deterministas, cualquier salida del sistema -si no existe intervención externa- será cíclica. Recordaréis que cada vez que he hablado de generación de claves a lo largo del taller, he sido bastante pesado con la cantinela de que un computador no puede generar números aleatorios, que debe existir adición de entropía externa al sistema por parte del usuario...

Sin embargo un computador sí puede generar números pseudoaleatorios en los llamados PRNG (del inglés PseudoRandom Number Generator). Éstos toman unos datos iniciales (denominados semilla) para, a través de una serie de complejas transformaciones, obtener una salida aleatoria. Si lo pensamos un poco, tiene cierto parecido a algo que ya conocemos... los algoritmos hash, pues en ambos casos la entrada no debe ser deducible a partir de la salida.

Efectivamente, los PRNG's y la criptografía van cogidos de la mano. De hecho, el mejor algoritmo de generación de números pseudoaleatorios que existe en la actualidad, está inspirado en uno de los algoritmos de cifrado de flujo más extendidos, diseñado por Ron Rivest (uno de los padres de RSA): RC4. Dicho algoritmo se llama ISAAC (Indirection, Shift, Accumulate, Add, and Count).

ISAAC fue ideado por Robert J. Jenkins en 1996, basándose en el generador de números pseudoaleatorios del algoritmo RC4 de Ron Rivest. A pesar de haber sufrido en 2001 un ataque criptoanalítico por parte de Marina Pudovkina que demostró la posibilidad de reducir la complejidad esperada para recuperar el estado inicial (10^2466) a una mucho menor (4,67 * 10^1240), no se considera la seguridad de ISAAC comprometida a efectos prácticos, dado que aún con el ataque de Pudovkina, sigue siendo computacionalmente inviable lanzar un ataque para recuperar el estado inicial del algoritmo, máxime cuando ISAAC suele ser aplicado en múltiples pasadas.

> *Si alguien desea saber más acerca del funcionamiento de ISAAC, así como de su relación con RC4, puede visitar la página oficial del algoritmo.*
>
> *http://www.burtleburtle.net/bob/rand/isaac.html*

Así pues, a partir de estos generadores de números pseudoaleatorios, se construyen algoritmos o métodos de borrado seguro. Vamos a ver los dos más importantes y más extendidos.

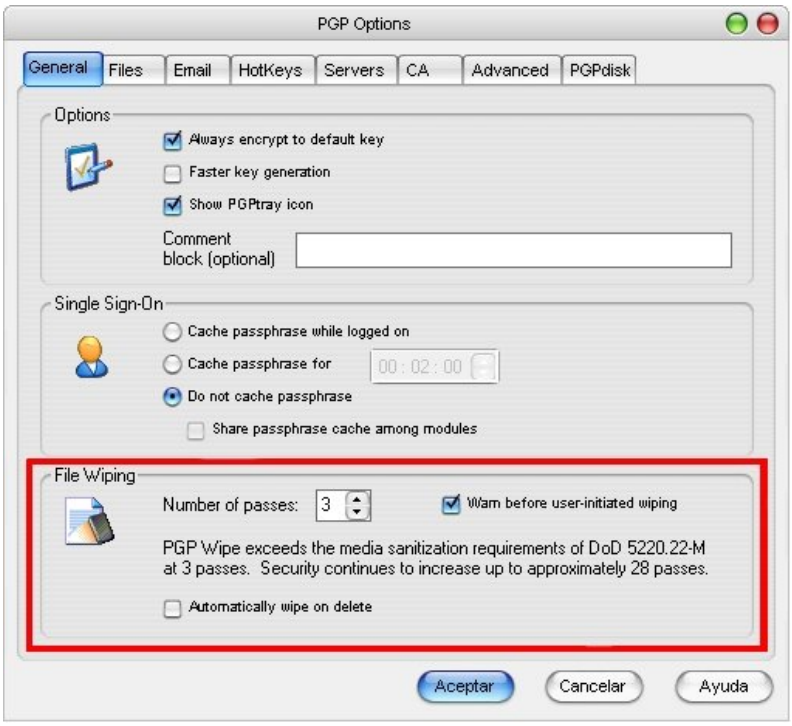
El primer método es el 5220-22.M del Departamento de Defensa de Estados Unidos. Este método fue diseñado para cumplir con el NISPOM (National Industrial Security Program Operating Manual), concretamente con la sección 8-306 (http://www.dss.mil/isec/change_ch8.htm). El método consiste en una serie de pasadas de sobrescritura (usualmente se consideran suficientes tres) con datos aleatorios generados con ISAAC.

El segundo es el llamado método Gutmann, ideado por Peter Gutmann. Este método se basa en 35 pasadas de sobrescritura de datos: las cuatro primeras con datos aleatorios (usualmente generados con ISAAC), las 27 siguientes tienen unos patrones fijos pero son aplicados en orden aleatorio, y las cuatro últimas son nuevamente datos aleatorios.

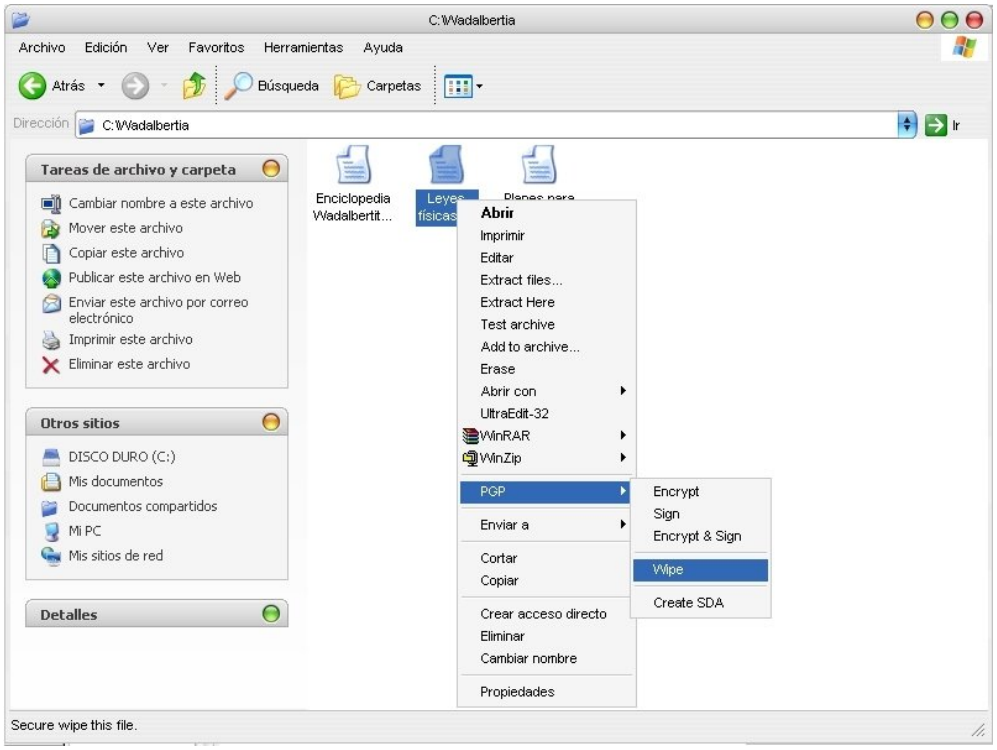
> *Para conocer a fondo el funcionamiento del método Gutmann y los patrones de sobrescritura de datos, es muy recomendable leer el paper en el que el propio Gutmann definió su método, titulado "Secure Deletion of Data from Magnetic and Solid-State Memory" y que podemos encontrar en el siguiente enlace:*
>
> *http://www.usenix.org/publications/library/proceedings/sec96/full_papers/gutmann/*

Existen otros métodos, como por ejemplo NAVSO P-5239-26 (RLL), NAVSO P-5239-26 (MFM), VSITR y GOST P50739-95, aunque ahora mismo los dos métodos que hemos visto son los más utilizados, y son considerados completamente seguros a nivel doméstico y empresarial. Sí es cierto que ciertas fuentes aseguran -con toda la razón del mundo- que el uso de patrones (como el método Gutmann) o la reescritura de pocas pasadas (como el DoD 5220-22.M) resulta poco recomendable en unidades comprimidas, por usar datos potencialmente altamente comprimibles que facilitarían la labor de recuperación de datos. Esta eventualidad puede solucionarse usando sobrescritura en múltiples pasadas con datos puramente aleatorios, lo cual elimina el elemento estadístico de datos comprimibles. Para un usuario doméstico no es necesario considerar hasta tal punto la seguridad en el borrado de datos... aunque seguro que habrá gente que prefiera considerarlo. ;-)

Los que hayáis seguido este taller desde el principio quizá recordéis el final del primer artículo, cuya temática era PGP y la introducción al sistema OpenPGP. Al final de dicho artículo hice una somera introducción a la característica de PGP que permite el borrado seguro de datos. Ahora que conocemos los fundamentos de la recuperación de datos, así como de su borrado seguro, la utilidad de borrado seguro de datos de PGP no tiene ningún secreto para vosotros. En las opciones de PGP podemos configurar el número de pasadas de sobrescritura que se llevarán a cabo cuando seleccionemos el borrado seguro de datos. La configuración por defecto está fijada en tres pasadas de sobrescritura, lo cual es más que suficiente para lo que vamos a necesitar. Es altamente recomendable activar la opción "Warn before user-initiated wiping", pues cuando seleccionemos borrar un fichero con la opción wipe de PGP, nos mostrará un aviso y pedirá confirmación. Puede ahorrarnos algún susto... más teniendo en cuenta que lo que borremos con wipe JAMÁS volverá. Por ese mismo motivo, no recomiendo activar la opción "Automatically wipe on delete".



La primera utilidad de wipe es borrar de forma segura uno o varios ficheros. Para ello, debemos seleccionar el o los ficheros que deseemos, pulsar con el botón derecho del ratón para abrir el menú contextual, seleccionar el menú de PGP y a continuación pulsar sobre wipe. PGP nos mostrará un diálogo con la lista de archivos a eliminar y nos pedirá la confirmación definitiva. Tras aceptar, esos ficheros desaparecerán para siempre de nuestro disco duro.

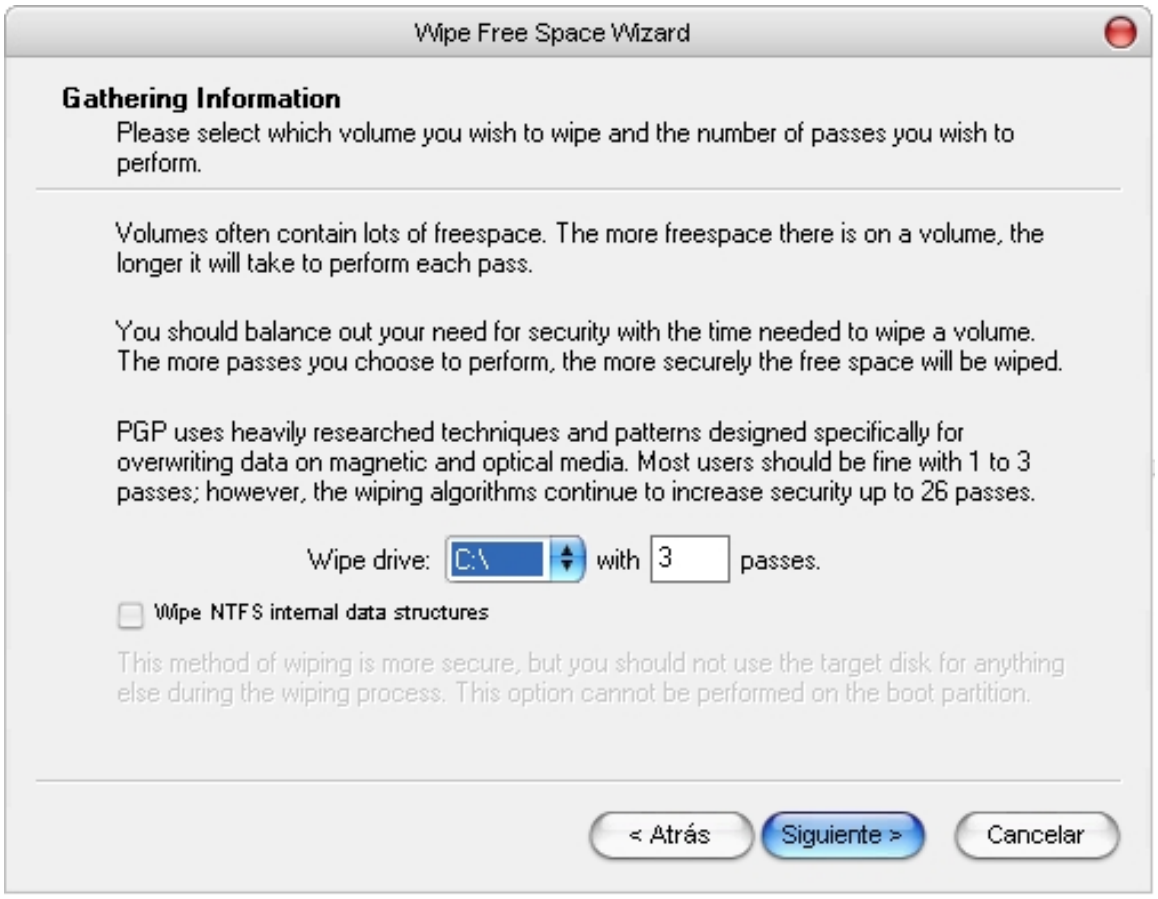


La segunda utilidad de wipe nos permite borrar de forma segura todo el espacio libre en una partición.

¿Borrar el espacio libre? Pues sí. Como hemos visto al trabajar con PC Inspector File Recovery, todo disco tiene gran cantidad de datos "perdidos" que pertenecen a archivos borrados de la forma tradicional y que se quedan en el disco de forma residual hasta que los sectores que ocupaban originalmente sean reescritos con nueva información.

Si no utilizamos siempre una herramienta de borrado seguro para eliminar los ficheros (y no creo que nadie lo haga... borrar de forma segura un par de gigas de vídeos puede llevar su tiempo), el disco duro tendrá siempre información residual en los sectores marcados por el sistema operativo como libres. Por ello, es altamente recomendable realizar de vez en cuando un borrado seguro del espacio libre en el disco.

Para llevar a cabo esta operación, seleccionaremos la utilidad PGPmail desde el icono de PGP en la bandeja de sistema y pulsaremos sobre el botón situado más a la derecha. Nos encontramos ante una pantalla de bienvenida a la utilidad "Wipe Free Space Wizard" de PGP, tras la cual encontraremos otra pantalla en la que debemos seleccionar la unidad a limpiar, así como el número de pasadas a realizar.



La opción "Wipe NTFS internal data structures" puede resultar útil a los usuarios del sistema de ficheros NTFS, pero OJO: no puedes realizar ninguna operación sobre el disco en el que quieras realizar un borrado seguro con esa opción. Por ese mismo motivo, no es posible utilizar esa opción de borrado en la partición de arranque (el propio PGP y el sistema operativo realizan constantemente operaciones sobre el disco).

A continuación nos encontramos con una pantalla donde se resume la operación que será llevada a cabo con los siguientes datos: sistema de ficheros, número de clusters, sectores por cluster, bytes por sector, capacidad total y unas barras de progreso que indican respectivamente el avance de la operación y la pasada actual. A la derecha del cuadro informativo hay dos botones: el primero (Begin Wipe) inicia inmediatamente la operación, mientras que el segundo (Schedule) permite programar la tarea para otro momento. Una vez iniciado el borrado, comenzará un escaneo completo del disco y la operación de limpieza propiamente dicha.

El Eliminador

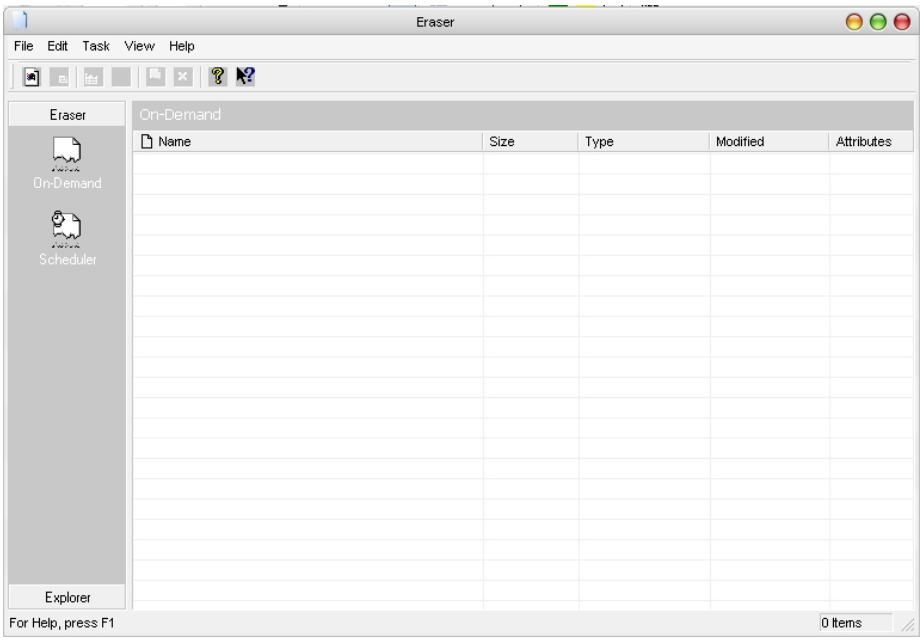
La utilidad de borrado seguro de PGP nos puede resultar muy útil, pero existe otra herramienta para sistemas Windows que es más eficiente en esa tarea. Dicha herramienta se llama Eraser y podemos descargarla desde su página oficial: http://www.heidi.ie/eraser/. Dado que se trata de software libre, podremos descargar la versión completa del programa con todas sus funciones.

Como siempre, comprobamos el tamaño y los hashes...

```
C:\WINDOWS\system32>dir "c:\Mis Documentos\Eraser57Setup.zip"
{...}

21/08/2005  14:31         2.811.211 Eraser57Setup.zip
{...}

C:\WINDOWS\system32>md5 "c:\Mis Documentos\Eraser57Setup.zip"
B30DADC346D2E3B2DECA9D84547BC03E  c:\Mis Documentos\Eraser57Setup.zip

C:\WINDOWS\system32>sha1 "c:\Mis Documentos\Eraser57Setup.zip"
NAME:  c:\Mis Documentos\Eraser57Setup.zip  HASH:
8d3b1c718a37587e5b0536ca1f29d369f329b58c

C:\WINDOWS\system32>
```

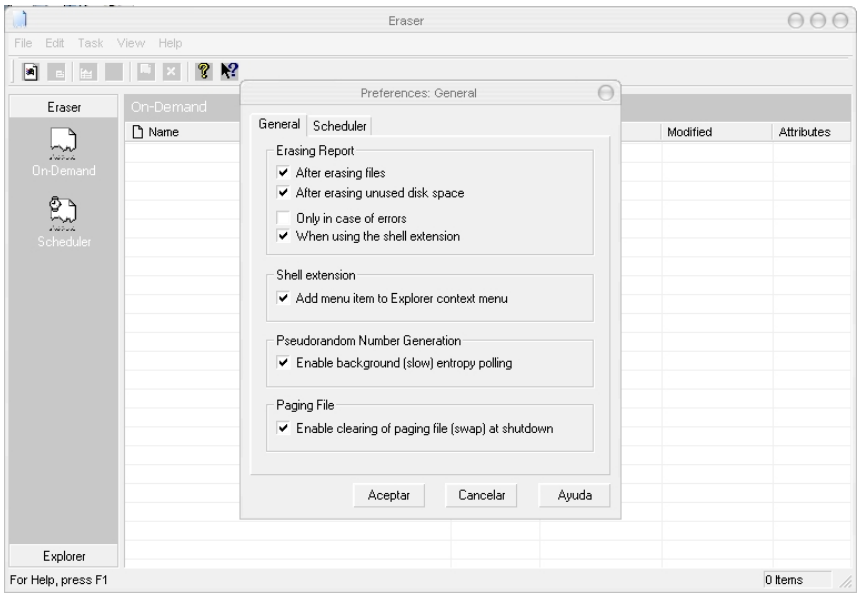
Una vez descomprimido en una carpeta temporal, nos encontramos con tres ficheros de texto (COPYING.txt, History.txt y README.txt), el instalador y su firma PGP. Para comprobar la firma, abrimos el fichero asc y seleccionamos el instalador cuando nos pida la ruta al fichero. Si no disponemos de la clave PGP apropiada en nuestro anillo, PGP se conectará al servidor de claves para buscarla. Tras importarla, podremos comprobar la validez de la firma.

Tras un proceso de instalación completamente normal (no creo que nadie tenga problemas con este punto), podemos comenzar a utilizar el software. Una vez iniciado el programa, nos encontramos con dos apartados en los cuales podemos asignar nuevas tareas: On-Demand (bajo petición) y Scheduler (programador). Ambas son autoexplicativas: la primera servirá para comenzar una tarea en el momento que el usuario lo solicite, mientras la segunda automatizará el momento de comenzar dichas tareas.



La asignación de nuevas tareas es muy sencilla: simplemente debemos elegir el tipo de tarea (entre limpiar el espacio libre en disco, una carpeta o un fichero) y si deseamos que ésta permanezca en la lista una vez finalizada. Adicionalmente, si estamos asignando la tarea al programador, deberemos fijar la periodicidad y la hora de inicio de dicha tarea mediante las opciones de la pestaña “Schedule”.

No es en el uso de Eraser donde vamos a centrarnos, pues además es bastante sencillo, sino en las opciones del mismo. Hay dos diálogos de opciones en Eraser: el primero de ellos para opciones generales y el segundo para opciones de borrado.



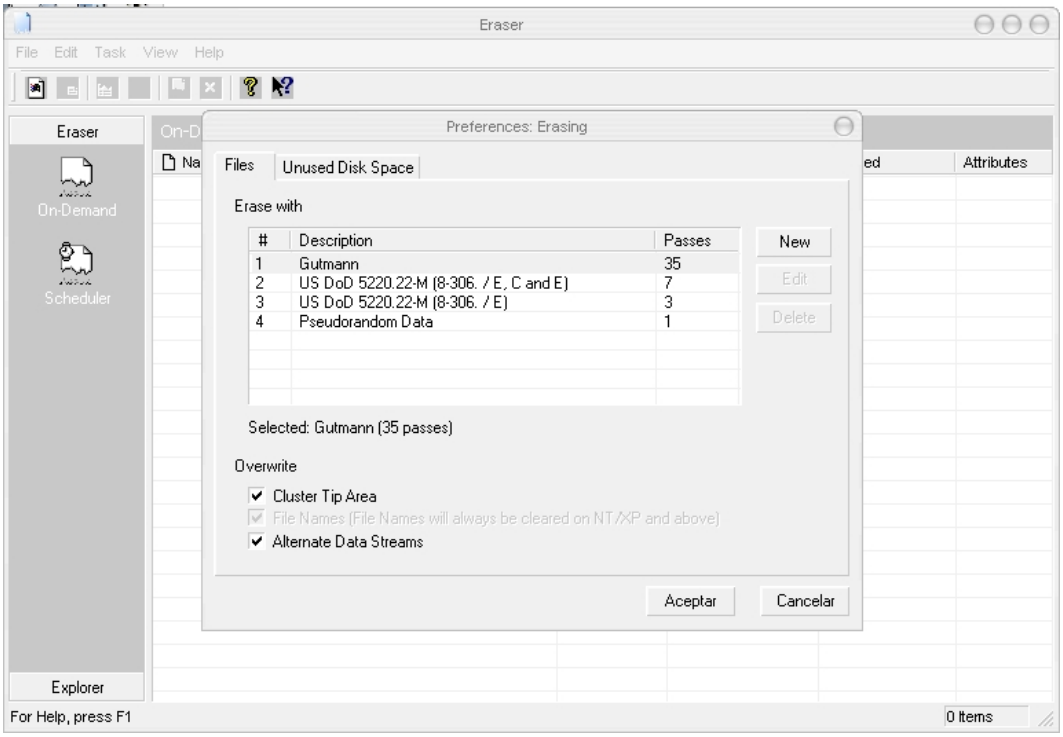
Accedemos a las opciones generales (Edit > Preferences > General, o bien mediante el atajo de teclado "Ctrl+P") y echamos un vistazo a las mismas. En primer lugar, tenemos la pestaña "General":

La primera sección (Erasing Report) configura los informes del programa. Podemos dejarlos como vienen configurados por defecto sin ningún problema, o podemos eliminar alguna opción si no deseamos demasiados informes. La segunda sección (Shell extension) sirve para habilitar el acceso rápido a Eraser mediante el menú contextual de la interfaz de Windows. Una vez habilitado, podemos acceder a él mediante el menú contextual del entorno gráfico de Windows en la etiqueta Eraser, y su funcionamiento es muy similar al de la utilidad wipe de PGP (nos pide confirmación antes de eliminar los datos) excepto por el detalle de poder seleccionar en ese mismo diálogo las opciones de borrado seguro (de las que hablaremos después). La tercera sección (Pseudorandom Number Generation) permite que el sistema de generación de números pseudoaleatorios (ISAAC) esté corriendo constantemente en segundo plano, opción que resulta muy interesante para aumentar la efectividad del mismo. La cuarta sección activa el borrado de la memoria swap (memoria virtual) al cerrar el sistema, aunque sólo es compatible con sistemas Windows NT.

> *¿Qué es la memoria virtual? Es un área de memoria secundaria (disco duro) que el sistema operativo utiliza como si fuera memoria principal (RAM). En sistemas Windows 9x (95, 98, 98SE y ME) se almacena en el fichero WIN386.SWP, mientras que en sistemas Windows NT (NT, 2000, XP y 2003) es el fichero pagefile.sys el encargado de esta tarea. En sistemas Linux existe una partición adicional encargada de almacenar la memoria virtual (partición Linux swap).*
>
> *Esta memoria es muy importante para que el sistema pueda mantener a mano grandes cantidades de datos ejecutados sin tener que saturar la memoria RAM, ni volver a generarlos. Además, permite opciones como la hibernación y/o suspensión de equipos portátiles.*
>
> *Pero tiene un lado negativo: en la RAM se almacenan gran cantidad de datos sensibles utilizados por programas en ejecución (por ejemplo, un número de tarjeta de crédito almacenado por el navegador web mientras realizábamos una compra, una contraseña de una aplicación en uso...) y estos pueden ir a parar a la memoria virtual en cualquier momento. Además, Windows no borra por defecto esta memoria virtual al apagarse o reiniciarse, con lo que dichos datos persisten en nuestro disco duro, suponiendo un evidente fallo de seguridad.*
>
> *Es posible forzar la eliminación del fichero de paginación de un sistema NT (mediante su completa sobrescritura. No es borrado seguro, pero es suficiente si no tratamos con aplicaciones críticas) mediante la creación de una clave en el registro de Windows. La ubicación de dicha clave sería "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management", su nombre "ClearPageFileAtShutdown", su tipo "REG_DWORD", y su valor "1". Así, cada vez que apaguemos o reiniciemos el sistema, el fichero pagefile.sys será eliminado.*

Esta opción, junto a una limpieza del espacio libre en el disco duro de forma regular, sería suficiente. Pero lo ideal sería cifrar la memoria virtual del ordenador, lo cual puede realizarse con ciertos programas como BestCrypt (http://www.jetico.com/index.htm#/bcrypt7.htm), que lamentablemente es de pago, motivo por el cual no será tratado en el presente artículo.

En la pestaña "Scheduler" podemos configurar las opciones del programador, todas ellas autoexplicativas. Puede resultar interesante, si se realiza un uso intenso del programador, activar su arranque automático al iniciar Windows. En mi caso, no la tengo activada porque prefiero realizar las tareas yo mismo.



Es el momento de echar un vistazo a las opciones de borrado (Edit > Preferences > Erasing, o bien mediante el atajo de teclado "Ctrl+E"). En la primera pestaña (Files) nos encontramos con las opciones de ficheros, donde podemos elegir el método de eliminación y otras opciones adicionales: borrar el área denominada cluster tip (el área restante sin utilizar en el último cluster del fichero); borrar los nombres de fichero (opción que siempre se realizará en sistemas NT); y borrar las ADS (Alternate Data Streams).

¿No sabes lo que es ADS? Digamos que es una característica del sistema de ficheros NTFS que la gente de Microsoft decidió incluir porque en HFS (un sistema de ficheros de Macintosh) daba buenos resultados. Pero como hay formas y formas de implementarlo, en Microsoft lo hicieron de una forma "poco buena", que dio por resultado una de las vulnerabilidades locales más graves de los actuales sistemas NT con sistemas de ficheros NTFS: los ficheros stream.

Los más veteranos seguidores de la revista quizá recuerden los ficheros stream del artículo de AZIMUT publicado en el número 6. Anda que no ha llovido...

Si queréis saber más sobre las ADS y los ficheros stream, echad un vistazo a este enlace:

http://en.wikipedia.org/wiki/Alternate_Data_Streams

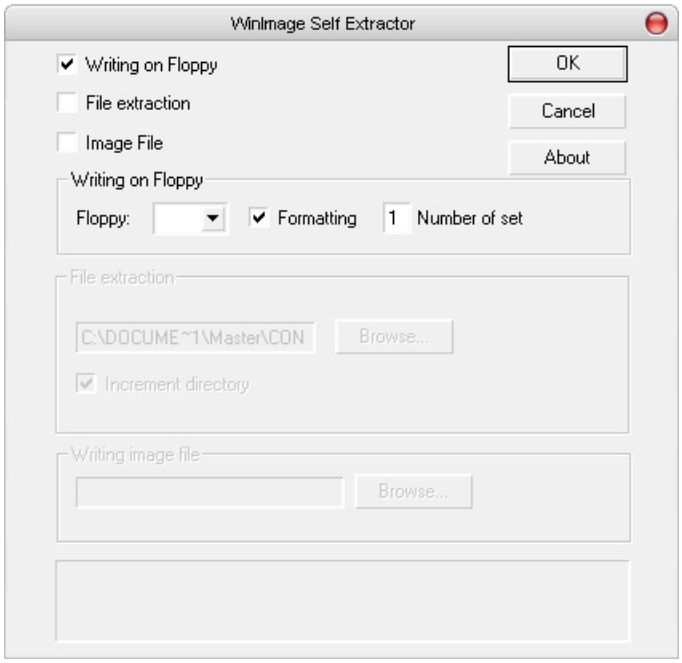
En la segunda pestaña (Unused Disk Space) podemos elegir el método de eliminación y las siguientes opciones: limpiar espacio libre (e índice), área cluster tip (en este caso, de cada uno de los ficheros en el disco) e información de directorios borrados.

Pero lo que más nos interesa es la configuración del método de eliminación, que es igual tanto para ficheros como para el espacio libre en el disco duro. Eraser trae por defecto configurados cuatro métodos de borrado seguro: el método Gutmann (del que ya hemos hablado), un método DoD 5220-22.M de 7 pasadas y otro de 3 (el estándar), así como un método de sobrescritura con datos pseudoaleatorios generados por ISAAC que podemos configurar para realizar desde una sola pasada hasta 65535 (para los más paranoicos ;-P).

Además, y ésta es la opción más interesante de Eraser, es posible definir un método completamente nuevo por nosotros mismos, de tantas pasadas como queramos e incluyendo patrones predefinidos o datos aleatorios. Gracias a esta opción, podemos construir, por ejemplo, variantes del método Gutmann, o aplicaciones sucesivas del mismo combinadas con otros métodos... las posibilidades son enormes.

Hay tres aplicaciones más de Eraser que vamos a conocer:

La primera, que resulta bastante curiosa, es la utilidad Eraser Verify. Con ella, podremos ver pasada a pasada el proceso de sobrescritura del fichero a eliminar. Las dos primeras veces resulta curioso, pero más allá no creo que lo uséis mucho.



La segunda es la utilidad Nuke Boot Disk. Basada en el proyecto Darik's Boot and Nuke (http://sourceforge.net/projects/dban/), nos permite generar un disquete de arranque de forma que al arrancar el ordenador con él, se realice un borrado seguro de TODO el disco duro. Muy útil para evitar situaciones como las que vimos en el estudio del principio del artículo, si quisiéramos deshacernos de nuestro disco duro.

La tercera y última utilidad extra de Eraser, muy poco conocida, es su versión de línea de comandos. Mediante ella, podremos realizar las mismas operaciones de borrado seguro que con su hermana gráfica.

```
C:\Archivos de programa\Eraser>eraserd /?
Eraser 5.6 for DOS. Free Software.
Copyright 2002 Garrett Trant. (http://www.heidi.ie/eraser/)

Usage:
  eraserd [Data] [-passes passes] [-silent]

  Data:

    -file       data [-nodel]
    -folder     data [-subfolders] [-keepfolder]
    -disk       drive: [-notips]
    -allfiles   drive:

  Parameters:
```

```
-file          Erase file(s) (wildcards allowed)
 -nodel          Do not delete file(s) after erasing
-folder         Erase all files in the folder
 -subfolders      Include subfolders
 -keepfolder      Do not delete the folder
-disk          Erase unused space on the drive
 -notips         Do not erase cluster tip area
-allfiles      Erase all files on a drive
-passes          Number of overwriting passes (default 1)
-silent        Nothing to standard output

C:\Archivos de programa\Eraser>
```

¿Y qué pasa con Unix?

Tranquilos, no me he olvidado de los usuarios de sistemas Unix. De hecho la mayoría de los sistemas Unix-like (como Linux y xBSD) incluyen de serie un comando de borrado seguro de datos: shred (puedes ver su manpage aquí: http://rootr.net/man/info/shred). Lo malo del comando shred es que únicamente sobrescribe la información que contiene el fichero, pero no lo elimina del disco, lo cual hace que tengamos que hacerlo nosotros mismos con el consiguiente trabajo extra. Por ello, usaremos otra herramienta para borrado seguro de datos en Linux: wipe.

La utilidad wipe (http://wipe.sourceforge.net/) es una herramienta de borrado seguro de datos que utiliza como generador de entropía /dev/urandom (o bien /dev/random cuando el anterior no está disponible), como generador de números pseudoaleatorios el algoritmo Mersenne Twister (http://en.wikipedia.org/wiki/Mersenne_twister) y el método Gutmann como algoritmo de borrado seguro.

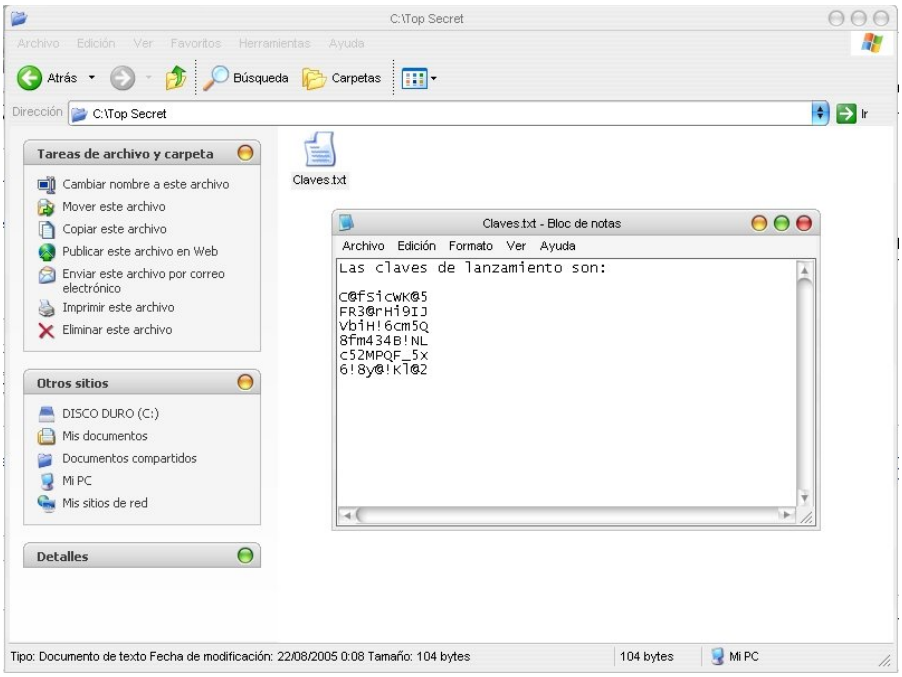
Su utilización es muy sencilla (eso sí, por defecto necesitamos privilegios de root para poder utilizarlo), veamos un ejemplo:

```
master@blingdenstone:~$
master@blingdenstone:~$ su
Password:
blingdenstone:/home/master# wipe prueba.txt
Okay to WIPE 1 regular file ? (Yes/No) yes
Operation finished.
1 file wiped and 0 special files ignored in 0 directories, 0 symlinks removed but not followed, 0 errors occured.
blingdenstone:/home/master# exit
exit
master@blingdenstone:~$
```

Y el fichero en cuestión ya no existe.

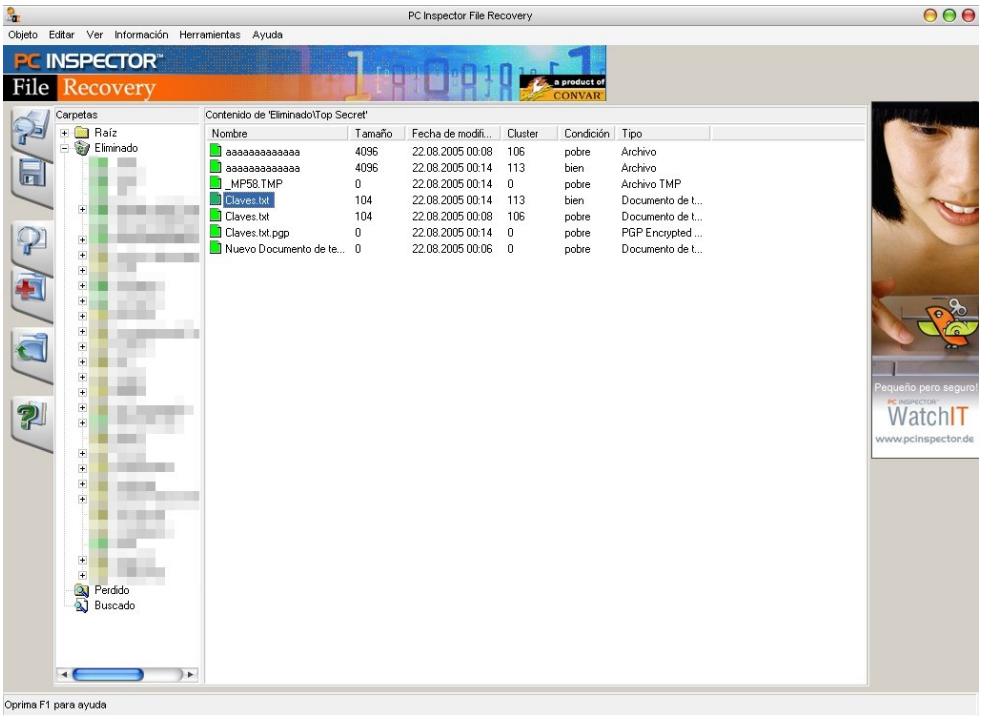
Los archivos secretos...

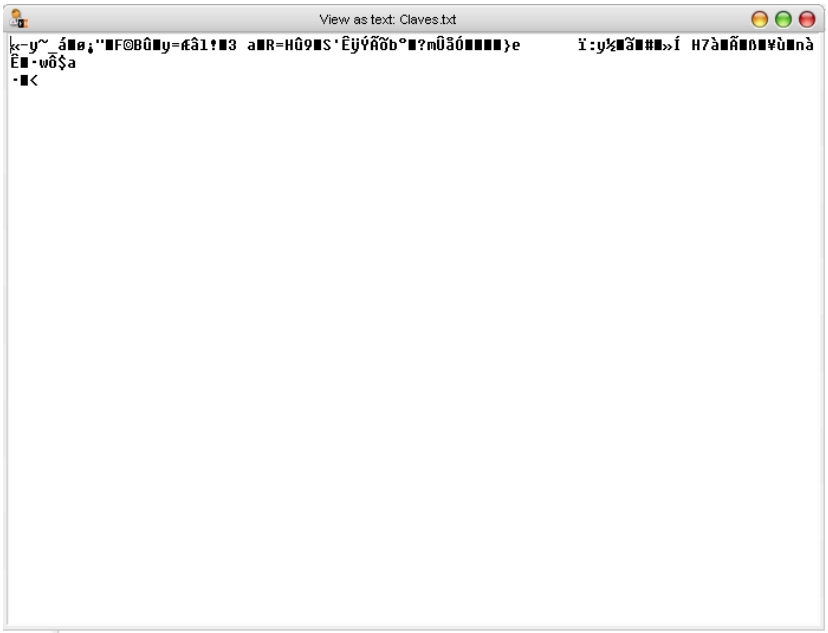
Con todo lo que sabemos ya sobre cifrado de ficheros y con lo que hemos aprendido hoy sobre borrado seguro de datos, ya somos perfectamente capaces de mantener los datos de un archivo absolutamente seguros de miradas indiscretas.



Tenemos un archivo con las claves de lanzamiento de los embudos en el directorio “Top Secret” (sí, para no llamar la atención...). Ciframos dicho archivo y lo borramos de forma segura (podemos usar la opción del diálogo de cifrado de PGP “Wipe original”), pero en ese mismo momento nos acordamos de que la última clave está mal. Desciframos el fichero, lo editamos corrigiendo el fallo, y volvemos a cifrarlo y a borrar de forma segura el original.

Ahora abrimos PC Inspector File Recovery para echar un vistazo a la carpeta en cuestión. De entrada, la presencia de ficheros del tipo “aaaaaaaaa” suele ser síntoma de un borrado seguro, pero vemos que hay varios ficheros: dos ficheros “aaaaaaaa” que no contienen ninguna información útil -procedentes del borrado seguro-, un fichero temporal vacío, un fichero “Nuevo...” de la generación del fichero de claves, un fichero cifrado vacío (y aunque no lo estuviera...), y dos ficheros “Claves.txt”... ¡eh! Uno de ellos tiene buena condición. Pues vamos a ver qué contiene. Como sospechábamos, no contiene nada útil porque aunque la condición de los datos que componían el fichero antes de borrarlo es buena, dichos datos no eran más que la última pasada del borrado seguro que sobre él efectuamos. No detallo el proceso en Unix porque sería idéntico cambiando PGP por GnuPG y el wipe de PGP (o Eraser, o...) por wipe.





Como vemos, gracias a la unión de las fuerzas de la criptografía y el borrado seguro, tenemos la posibilidad de almacenar datos absolutamente seguros en nuestro sistema. Es recomendable, también, realizar una limpieza de espacio libre en disco periódicamente para mantener el sistema en óptimas condiciones.

Existen, además de todos los citados hasta ahora, multitud de programas de borrado seguro de datos que quizá os puedan interesar: Bcwipe (http://www.jetico.com/bcwipe3.htm), MindSoft Wipe (http://www.mindsoftweb.com/productos/wipe.htm), R-wipe & Clean (http://www.r-wipe.com/), East-Tec Eraser (http://www.east-tec.com/eraser/index.htm), BiteByBite (http://www.potentialsys.com/Default.aspx?Page=BiteByBite), Ultra Shredder (http://www.xtort.net/xtort/ultra.php), Drive Scrubber (http://www.iolo.com/ds/index.cfm), Sure Delete (http://www.wizard-industries.com/sdel.html), Shred XP (http://www.wizard-industries.com/sdel.html), File Shredder 2000 (http://www.gregorybraun.com/Shredder.html), Mutilate File Wiper (http://mutilatefilewiper.com/)... y si buscáis en Google encontraréis muchos más, os lo aseguro.

Cerrando la Caja de Pandora

Poder mantener datos seguros mediante el cifrado y eliminación segura de ficheros está muy bien para guardar unos pocos ficheros, por ejemplo un fichero de contraseñas o un log. Pero cuando se hace necesario mantener este proceso sobre una gran cantidad de ficheros (por ejemplo sobre el código completo de una web), se vuelve lento y engorroso. Para esas circunstancias estaría mejor poder disponer de una “caja mágica” cuyo contenido siempre fuera seguro y que nosotros únicamente pudiéramos abrirla y echar un vistazo dentro.

Pues efectivamente, esas “cajitas” estarían bastante mejor... y tienen un nombre: volúmenes virtuales cifrados.

Supongo que no habrá nadie que no sepa lo que es una imagen de CD o DVD: un fichero que contiene toda la estructura de directorios y ficheros de un medio concreto. Existen también utilidades (como las famosas Daemon Tools) que permiten manejar dispositivos virtuales que montan esos ficheros como si de unidades físicas se tratase. Pues se puede hacer exactamente lo mismo con discos duros: generar un fichero que contenga un disco duro virtual. Y, como cualquier otro tipo de fichero, éste puede estar cifrado en nuestro disco... y al montarlo con la llave adecuada, toda su información se volverá visible.

PGP incorpora dicha funcionalidad bajo el nombre PGPdisk en sus versiones desktop y corporate. Las versiones freeware no han incorporado nunca PGPdisk, pero eso no significa que no podamos usar PGPdisk... ;-)

Podemos usar la última versión de PGPi (la internacional) que tenía soporte para PGPdisk: la 6.0.2i. Podemos descargar dicha versión (compatible con NT) en este enlace: ftp://ftp.pgpi.org/pub/pgp/6.0/6.0.2i/PGPfreeware602i.exe.

```
C:\WINDOWS\system32>dir "c:\Mis Documentos\PGPfreeware602i.exe"
{...}

22/08/2005  02:07        6.859.666 PGPfreeware602i.exe
{...}

C:\WINDOWS\system32>md5 "c:\Mis Documentos\PGPfreeware602i.exe"
7F567514CF49531D5D631F1D6A8E51B7  c:\Mis Documentos\PGPfreeware602i.exe

C:\WINDOWS\system32>sha1 "c:\Mis Documentos\PGPfreeware602i.exe"
NAME:  c:\Mis  Documentos\PGPfreeware602i.exe  HASH:
99e97ccbcaff4206040e954fb710977c445943fb

C:\WINDOWS\system32>
```

Como en la propia página de PGPdisk de PGPi dice (http://www.pgpi.org/products/pgpdisk/), es posible compilar la versión 6.5.1i con soporte para PGPdisk, pero no será necesario porque tenemos otra opción mucho mejor... PGP CKT IMAD'S.

> *Como muchos sabréis, la versión 6.5.8 de PGP supuso un punto de inflexión en el mítico programa. A partir de dicha versión, NAI (Network Associates, dueña de PGP hasta Febrero de 2002) cambió bruscamente su política, lo cual derivó en que la última versión "libre" de PGP fuera la mítica 6.5.8. Con la adquisición de PGP Corporation de los derechos de PGP parecía que las cosas iban mejor... hasta la versión 9. :-(*
>
> *Por cierto, el enlace dado en el artículo anterior para adquirir PGP libre a través del MIT ya no es válido: el MIT -otro de los baluartes de PGP en tiempos- ya no distribuye PGP. :-(*
>
> *Pues bien, dado que dicha versión 6.5.8 era aún libre, se desarrollaron modificaciones sobre ella para adaptarla a los nuevos tiempos. No fueron muchas, dada la dificultad de trabajar con un software tan complejo como PGP, pero una de ellas es especialmente buena: la versión de IMAD'S.*

Conseguir la versión de IMAD's de PGP es algo complicado, pero podréis descargar sin problemas la beta3-build9 (crucemos los dedos para que de aquí a que salga la revista no pase nada...) desde el siguiente enlace: ftp://ftp.zedz.net/pub/crypto/pgp/pgp60/pgp658_ckt/pgp658ckt09b3.zip.

```
C:\WINDOWS\system32>dir "c:\Mis Documentos\pgp658ckt09b3.zip"
{...}

22/08/2005  02:24        4.157.320 pgp658ckt09b3.zip
{...}

C:\WINDOWS\system32>md5 "c:\Mis Documentos\pgp658ckt09b3.zip"
F10D9F6320570E18C14AF328204A13CB  c:\Mis Documentos\pgp658ckt09b3.zip

C:\WINDOWS\system32>sha1 "c:\Mis Documentos\pgp658ckt09b3.zip"
NAME:  c:\Mis  Documentos\pgp658ckt09b3.zip  HASH:
0b270c80a92b921bd54e8f7ca8626dd87e3cde2

C:\WINDOWS\system32>
```

También, por supuesto, podéis usar alguna versión de pago de PGP para utilizar PGPdisk, al igual que podéis utilizar algún otro tipo de software comercial que realice esta función, como Steganos Safe (http://www.steganos.com/?product=safe8&language=en&layout=web2005) o BestCrypt (http://www.jetico.com/bcrypt7.htm)... pero tampoco se acaban ahí las opciones. ;-)

TrueCrypt

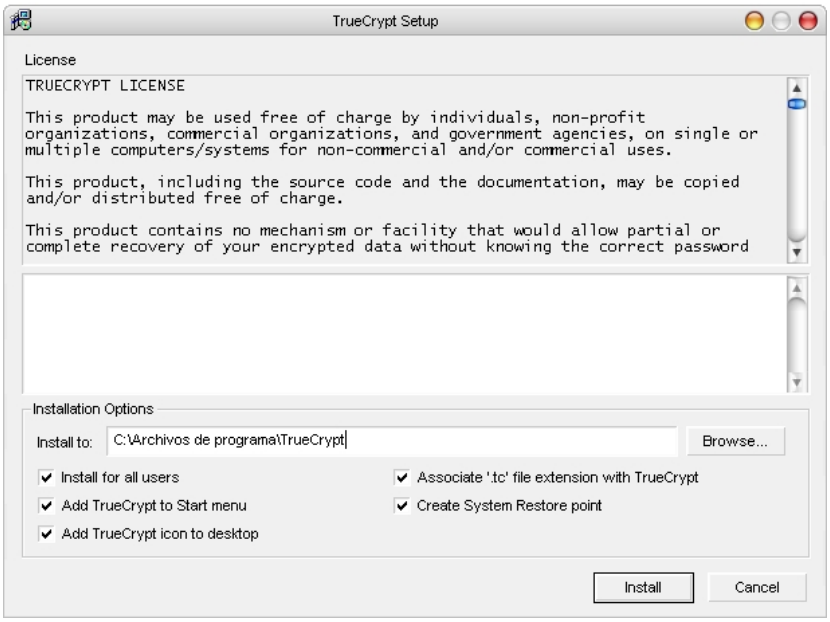
Existe un programa que es capaz igualmente de manejar volúmenes cifrados virtuales, pero que además es compatible con las nuevas versiones del sistema operativo Windows, es software libre y es completamente gratuito. ¿Se puede pedir más? Pues sí: unas opciones de cifrado realmente impresionantes.

Lo primero que vamos a hacer es visitar la página web de TrueCrypt (http://www.truecrypt.org/) y descargar la última versión (http://www.truecrypt.org/downloads.php), la 3.1a en el momento de escribir estas líneas: http://www.truecrypt.org/download.php?file=truecrypt-3.1a.zip. Además, descargaremos también la firma PGP (http://www.truecrypt.org/download.php?file=truecrypt-3.1a.zip.sig) y la clave pública con la que fue generada (http://www.truecrypt.org/downloads/TrueCrypt_Foundation_PGP_public_key.asc).

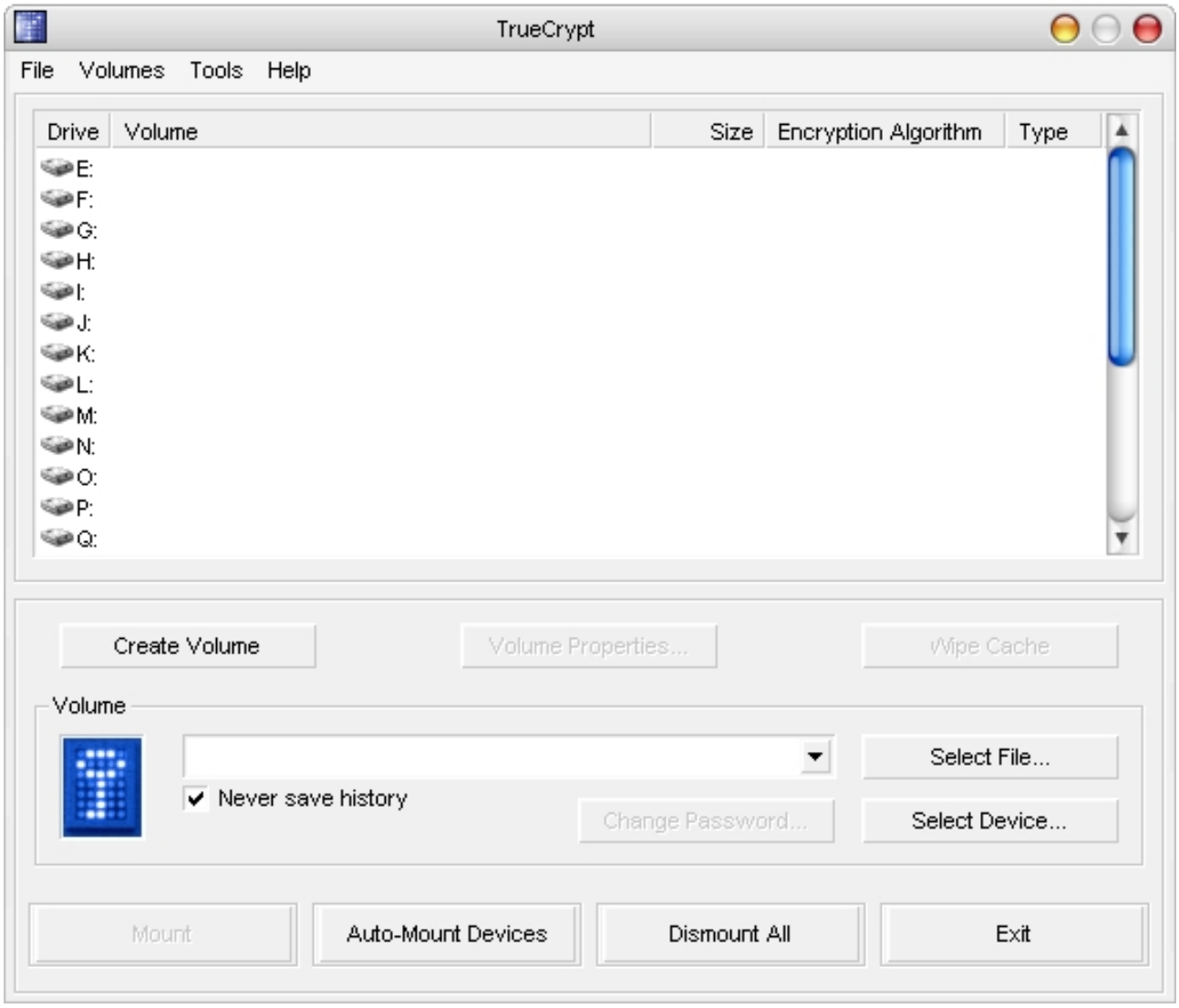
Tras comprobar la firma y -como siempre- tamaño y hashes...

```
C:\WINDOWS\system32>dir "c:\Mis Documentos\truecrypt-3.1a.zip"
{...}

22/08/2005  02:32           655.539 truecrypt-3.1a.zip
{...}

C:\WINDOWS\system32>md5 "c:\Mis Documentos\truecrypt-3.1a.zip"
D2C031D3D3919F72B4F28B8ACF306BEB  c:\Mis Documentos\truecrypt-3.1a.zip

C:\WINDOWS\system32>sha1 "c:\Mis Documentos\truecrypt-3.1a.zip"
NAME: c:\Mis Documentos\truecrypt-3.1a.zip HASH: 40b7d635fc53335b627ac7081bf2637
1bb9bd2fd

C:\WINDOWS\system32>
```

... pasamos a instalar el software. El instalador es muy sencillo, con una única pantalla que nos muestra la licencia y nos pide la ruta de instalación. También nos muestra unas opciones de instalación, entre las cuales recomiendo encarecidamente dejar marcada la de “Create System Restore point”.

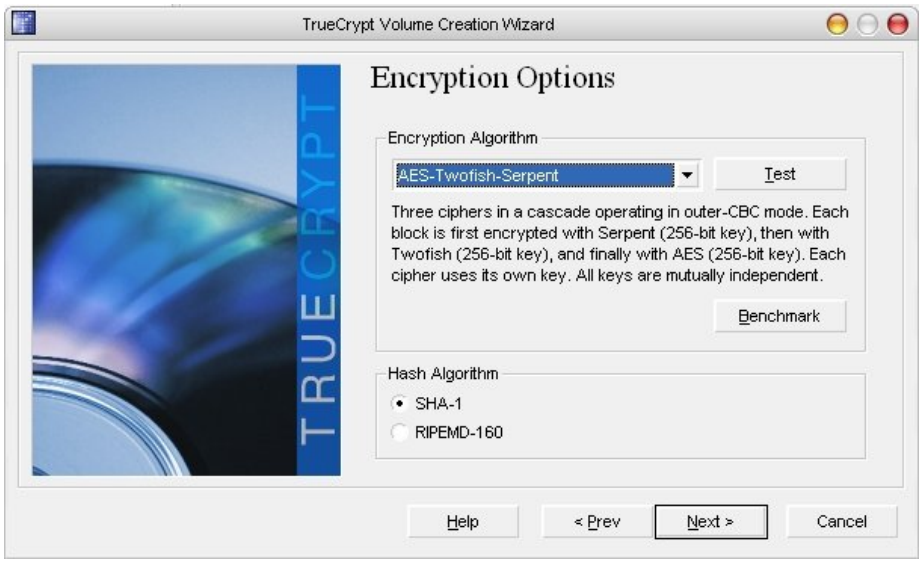


Al iniciar el programa nos encontraremos con la ventana principal que por el momento está bastante vacía. Es el momento de ponernos manos a la obra y crear nuestro primer volumen.



Al pinchar en el botón “Create Volume”, nos encontramos con el asistente de generación de volúmenes. La primera elección a realizar es si deseamos un volumen estándar o uno oculto. Un volumen oculto sería un meta-volumen virtual que se encuentra oculto en el espacio libre de un volumen virtual que le sirve como tapadera. Como la ayuda del software indica, esto puede resultar útil en situaciones en las que puedas verte forzado a revelar contraseñas y/o datos privados (la ayuda del programa menciona la situación de que nos lo pidan mediante la fuerza... pero seguro que estarían más bien pensando en peticiones judiciales :-m). Por el momento, con un volumen estándar nos sirve.

En el siguiente paso, se nos solicita la ruta al fichero donde deseamos guardar la información del volumen virtual. Adicionalmente, es posible pulsar en “Select Device” para cifrar particiones o discos completos; lo cual puede resultar muy útil, por ejemplo, para tener un pen-drive cuyo contenido esté permanentemente cifrado. Nosotros vamos a crear un volumen virtual normal en la ruta C:\master.tc.



La siguiente pantalla es una de las más interesantes. En primer lugar, debemos seleccionar el algoritmo de cifrado del volumen. Pero este algoritmo no tiene porqué ser único, sino que pueden utilizarse varios algoritmos en cascada para realizar sucesivos cifrados a la información en el volumen virtual. Yo seleccionaré una cascada AES-Twofish-Serpent.
También debemos elegir el algoritmo de hash utilizado, en mi caso SHA-1.

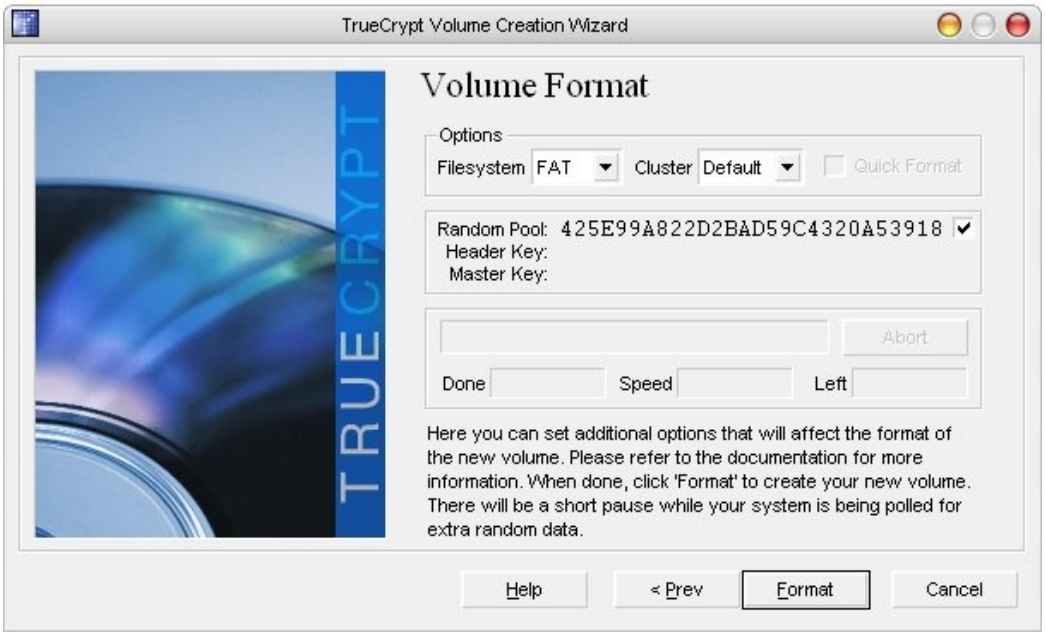
El algoritmo (o algoritmos) de cifrado utilizados para el volumen virtual condicionará la velocidad de la misma. Dependiendo de la complejidad del método de cifrado seleccionado y de la potencia de nuestra máquina, este volumen trabajará más deprisa o más despacio.

Para hacernos una idea de la velocidad con la que podremos trabajar, podemos pulsar en el botón "Benchmark" y realizar las pruebas pertinentes.

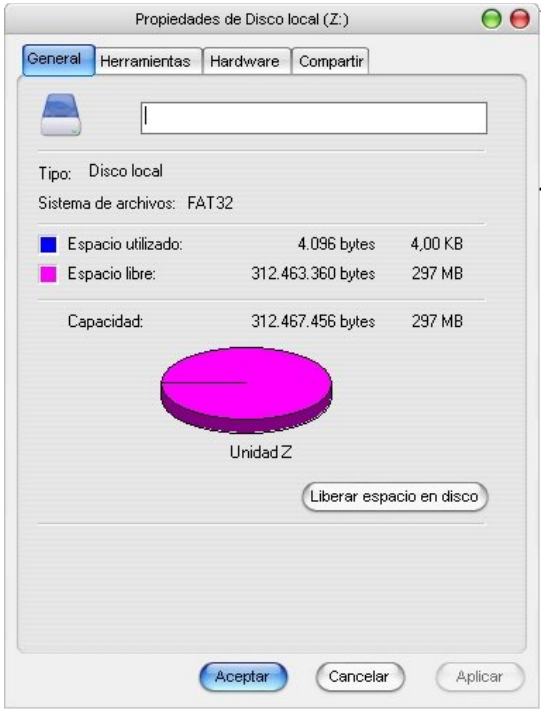
Tras seleccionar los algoritmos, se nos pedirá especificar el tamaño del fichero que hospedará el volumen virtual. En la siguiente pantalla deberemos introducir -por duplicado, como de costumbre- el password.

A continuación se procederá al formateo del volumen virtual, en el sistema de ficheros y tamaño de cluster especificado. Además, podremos ver cómo varía la semilla aleatoria. :-P

Tras formatear el volumen virtual, podemos regresar a la pantalla principal del programa y seleccionar el fichero generado para proceder a su montado en la letra de unidad seleccionada. Una vez proporcionada la contraseña para descifrar la partición, estamos en disposición de utilizar la unidad como un disco duro más en nuestro sistema.



Por desgracia, no existe ningún software similar -que yo sepa- para Unix, si bien en los planes de futuro de TrueCrypt (http://www.truecrypt.org/future.php) está el desarrollo de una versión para Linux.

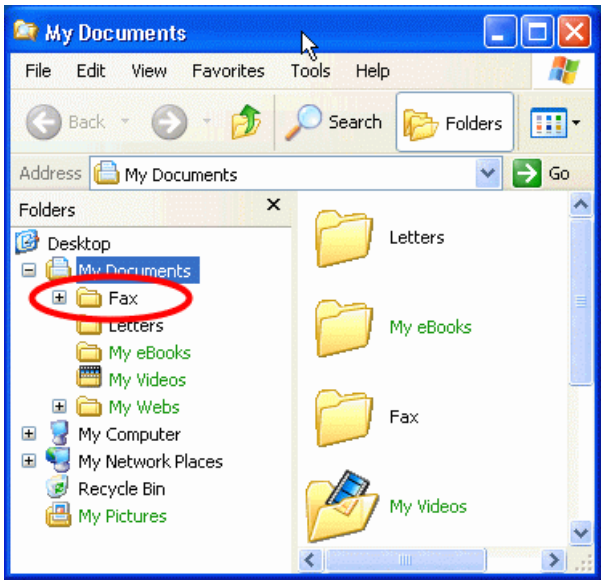


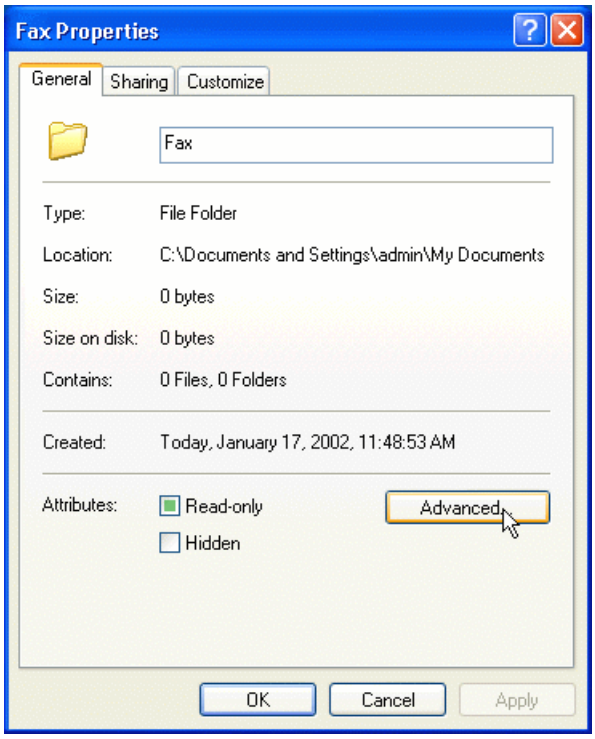
Más allá: sistemas de ficheros cifrados

La creación de volúmenes virtuales cifrados (o incluso el cifrado de una partición completa) tiene una pega: dependemos de un software, bien sea TrueCrypt, PGPdisk... pero es posible utilizar particiones cifradas (incluso volúmenes virtuales en algunos casos) mediante los llamados sistemas de ficheros cifrados.

Ya sabemos qué es un sistema de ficheros, así que imaginarnos un sistema de ficheros cifrado no es muy difícil, ¿verdad? Se trata de un sistema de ficheros que almacena la información en el disco cifrada en lugar de en la forma habitual.

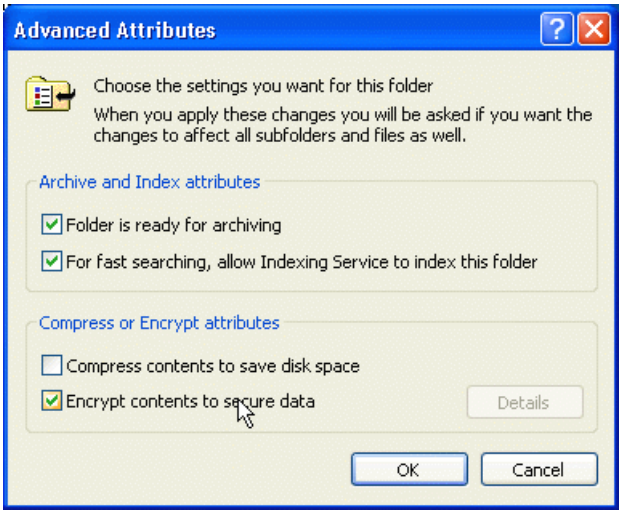
En Windows (en sus versiones 2000, XP Profesional y 2003) existe algo que podríamos denominar sistema de ficheros cifrado, aunque estrictamente no lo sea. Se trata de EFS (Encrypting File System), y mediante él se puede realizar cifrado y descifrado transparente de ficheros mediante la shell gráfica de Windows. Si queremos cifrar una carpeta, debemos acceder a sus propiedades mediante el menú contextual y seleccionar la opción "Atributos Avanzados" en la cual podremos activar el





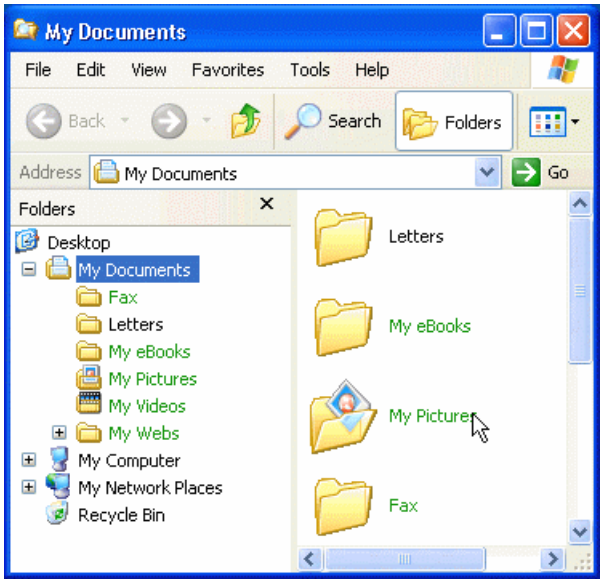
cifrado de la información, tras lo cual la carpeta aparecerá en color verde indicando así su condición.

Sin embargo, aunque conceptualmente es un gran avance respecto a programas como TrueCrypt, en la práctica EFS no supone ninguna ventaja sino en todo caso inconvenientes:



- No participamos en el proceso de generación de las claves, éstas se generan solas. A mí las claves que brotan de la nada no me gustan.

- El proceso de copia de seguridad de dichas claves para poder seguir utilizando dichos datos en caso de fallo del sistema es un suplicio: http://support.microsoft.com/default.aspx?scid=kb;EN-US;q241201. Es cierto que existen programas que pueden recuperar datos cifrados con EFS, como Advanced EFS Data Recovery (http://www.elcomsoft.com/aefsdr.html)... pero no sé qué sería peor, si que el software fuera muy lento e ineficaz en su tarea -y, por tanto, inútil-, o que fuera rápido y eficaz -y, por tanto, inútil EFS-.



- En realidad no es un sistema de ficheros propiamente dicho, pues necesita tener por debajo a NTFS como sistema de ficheros real.

Por todo ello, yo particularmente prefiero la utilización de volúmenes virtuales cifrados con, por ejemplo, TrueCrypt.

Pero existen otras implementaciones mucho mejores de sistemas de ficheros cifrados que resultan realmente muy útiles. En sistemas Linux (y otros sistemas Unix-like) existen sistemas de ficheros cifrados como EncFS -Encrypted File System- (http://arg0.net/wiki/encfs), CFS -Cryptographic File System- (http://www.crypto.com/papers/cfs.pdf), TCFS -Transparent Cryptographic File System- (http://www.tcfs.it/), StegFS -Steganographic File System- (http://sourceforge.net/projects/stegfs), CryptFS (http://citeseer.ist.psu.edu/zadok98cryptfs.html)...

Incluso existen otros sistema de ficheros cifrados buenos para DOS/Windows, como SFS -Secure File System- (http://www.cs.auckland.ac.nz/~pgut001/sfs/).

También es posible construirse en Linux un sistema de ficheros cifrado "a medida" gracias al soporte nativo del kernel para dispositivos loopback. Junto a la gran cantidad de soporte criptográfico que ofrece el núcleo de Linux, es posible construirse una partición cifrada con el algoritmo de cifrado que queramos: http://www.linuxsecurity.com/docs/HOWTO/Encryption-HOWTO/Encryption-HOWTO-4.html.

La ventaja evidente de uso de este tipo de sistemas de ficheros es la transparencia hacia el usuario del cifrado, que es un arma de doble filo, pues es a su vez su mayor debilidad: al ser transparente al usuario, es muy común perder las claves en un fallo de hardware, junto con toda la información guardada en el disco.

Ahora ya podemos decir que sabemos cómo blindar nuestro disco duro.

Hasta aquí la quinta entrega del Taller de Criptografía. Como siempre, si tenéis cualquier duda o problema con el artículo o con sus prácticas, o simplemente para cambiar impresiones o charlar un rato, os recomiendo que os paséis por el foro de la revista: http://www.hackxcrack.com/phpBB2/index.php.

Hasta la próxima. ;-)

Distribución de este documento

Este documento se distribuye en formato PDF realizado bajo OpenOffice.org 2.0.

Ficheros a distribuir:

Nombre: “tall_cript.pdf”
Descripción: Documento principal.

Nombre: “tall_cript.pdf.sig”
Descripción: Firma digital PGP del fichero “tall_cript.pdf” realizada por el autor.

Nombre: “hash.txt”
Descripción: Contiene las cadenas hash MD5 y SHA-1 de los ficheros “tall_cript.pdf” y “tall_cript.pdf.sig”.

Datos adicionales:

Las cadenas de hash MD5 y SHA-1 pueden resultar útiles para comprobar la integridad del fichero descargado, pero no son garantía de la inalterabilidad del documento, pues éste puede haber sido alterado junto a las cadenas de hash.

Para comprobar la completa autenticidad e inalterabilidad del fichero, es necesario utilizar el sistema OpenPGP para validar el fichero .sig (MIME/PGP) de firma. Cualquier modificación no autorizada del documento hará que la firma del mismo no sea válida, y ésta es imposible de falsificar.

Death Master

Autentificación:

-----BEGIN PGP MESSAGE-----
qANQR1DBwUwDJoT5ygJgu7ABEACgyFIvMIVJgZaxEaYUekeH8rYH+7QPzQo6ndRi
u7i4fOiWtFGgA+x3Aek9tR6qLMsMgdbbgCQKs/wyicqsqLhbwqoo6cJJw/VPgO7r
+JAqHORUmTG+rgAgHEZ8aIoD2lcNMZqXmArmkfbL3mDmmR9i8Q8yD5b1oCU4ZbYf
ETMX9Yj3FZFPdYm6TFxuqKgsuHCVoRJikBl64JAqYrctchGkAZWJZgYcB7iHPXk4
hM2XbAxEqBFKcRQG1dhdIDaQaE/n7AIlY59lVCyhurEgiiTmIPZ5YhyEgLBXyr4E
e6pr1n5HXPLnQ/DADP3VHBN6vGZHggNINJfteQ/yyl8GHihOZ/XEqRlWFHq0G9Hd
WakdlCvsLONY+AdE+lWKb9sSJP8ct+MZnB4XfTc78Lyg2w1EHSO7FDgAQa8vXXcV
TXhTWc5lppE72TSPPy8VGQ+nEciOMqWgnqm7lL2+/DzTjgUjyPOMtCZgLhLQhmy+
XoQuyJFxIKIpN/I+yShPlyDcCvyKEmQfZikLmfhyVv6jQHjgjsF8BChr42N220mp
Sarf9jkbcGK+rkXV4BKRnhrBBs9gzNDfopiAW6pn7CONbUknBzrHk6YtJQ+uc+qw
2dfRZ/zAaNWbGKeT6N++hsApg+Z5ltM+nTwk0AROjGEkFKTnCLvXoPsfis4aSwlM
qJiiedLDDgHPNupFDxxu2ef0+RuUlGyrEZdaXH5K87y6UvkMc5h4sbZk6wZTPifM
mJScFjv5xKV6krd+UueL+hGCz7RlbfuImVEyDDHZJKf0YH9y/7SE7wSDFPOO4r08
uN1C3RiqsvzOi3vm5V5rBVhVBqPQxyfDs91PPfwAfyS6vdqrlkpnz+OouKyoAEXP
ZCHEvCf1XFFgTj2tWHga7hAC9cGdxkqzlo+SajunT6mtNEj2cnk+y2j8zAHRkdA6
Qt63KpKFyaSXRtCxNUChTXQQGpukByE/hRG0udexi90cdmL23sPrpR1JvV0VQxbz
lZWJkgsyjAZ+Koq9xbvWdBMItpUvJjmBjpVFXZin0C/DzKg92ODhbor6/oB9U829
c/1XTLaghiD2ifvmH6CJsDkWMpQmyWqbOcMX3iSEYpVEI02D0kavNcT1Gm/l/NjN
ka3QpeJFjReGKJRQrvwJHbkOBfo/Ckb7sAnzJwX/RiDP5Bftbp8TcI9UrjW8KR+6
HDJfBJ2UOY1/rC6FkkX2qDZzN7pi03Ww7mSF2bauiQQbH+4sTgRoqNoh2VHbHYT5
tcNTo3/+CICArrOZwm7GzPxP/rw8oFVfRaZazkGwrCEl2eMKjnacKJHiPnoNCZiE
7cv6ksar90iywtWUBzKS9rJDJYBEgmsor744jQAWykcXe+4vxoSyEoqziDYiG3kv
cbXAsz8bC+N3BGvp56OKDQ9Gpo64XPjGbpLuJbLSEcTJZdMNkdKkjStqHALj6CYD
xe5OU/jd1VSPJzpwqxNKg7xI5guzwflFGEVQ4TRb3HDTZaexxRI95b/h4qYcuBN7
/ZK8j+PNCaw6ZSqeWiwmt2Mj2bBZds3s/7eZy0Rybei64JLXnM0nt0vrBRI+pDzY
p+MPM2cKvZX1qJS7qJ9yvQDE+Yylm2XcYaKqRY3WmmShJiOEY5sSufnqDoxguvEl
h25gYOF2gPj2PFjlpBdnHCLyAYxZZUo7epJI179qTvWwMX02CYDEsujZ5lzFcOVZ
DDb0fyoG7Qcm/XBv1VvKy3lj1jyrcrct9bCUKevFivd3w87kyDrJ+As5PFf7aevQ
kjWSOm7g5ZacQoyZQiuS1zAoIQKUUTwVny+TELXcPZBTrPXBUSWXuh0E89fkf0Xp
P1SBo3f9c9x3MGFpZBFxdbM4Ys8OXt3W6tzUkaDmcHtlVH43GnkUsDtepnOaUxiO
3YIckgujtLUT9vw/tfdNzaHnbQSkyAS2S/sdZU8aKohHWbd2uVHzAbMxCtto1Psg
rmKKm/2Kb/kKwy+VI1HhGl/C4JWJ
=jtmg
-----END PGP MESSAGE-----

Licencia

Taller de Criptografía – http://www.death-master.tk/ – Versión 1.0

Este documento ha sido liberado por su autor bajo la licencia GNU Free Documentation License (GFDL), y su utilización, copia o reproducción queda sujeta a los términos de la citada licencia, que puede ser consultada en el siguiente sitio web:

- **GNU Free Documentation License:** http://www.gnu.org/copyleft/fdl.html

GFDL Version 1.2, November 2002
Copyright (C) 2000, 2001, 2002 Free Software Foundation, Inc.

Copyright (c) 2005 Death Master
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with the Invariant Sections being “Distribución de este documento” and “Licencia”, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Cualquier copia, modificación, distribución o utilización en general de este documento debe respetar la autoría original del mismo, correspondiente a **Death Master**.

Cryptography Workshop – http://www.death-master.tk/ – Version 1.0

This document has been freed by its author under the license GNU Free Documentation License (GFDL), and its use, copy or reproduction is subject to the terms of the mentioned license that can be consulted in the following website:

- **GNU Free Documentation License:** http://www.gnu.org/copyleft/fdl.html

GFDL Version 1.2, November 2002
Copyright (C) 2000, 2001, 2002 Free Software Foundation, Inc.

Copyright (c) 2005 Death Master
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with the Invariant Sections being “Distribución de este documento” and “Licencia”, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Any copy, modification, distribution or general purpose use of this document should respect the original responsibility of it, corresponding to **Death Master**.

*** End Of File ***