

El Arte de la Esteganografía

Patricia Cardona y Guillem Cantalops

2000.02.14

Índice General

1	Introducción	3
1.1	História de la esteganografía	3
1.1.1	Los orígenes	3
1.1.2	El Manuscrito Voynich, un misterio sin resolver.	4
1.1.3	Esteganografía: desde el origen de la humanidad hasta nuestros días	5
1.2	Qué es realmente la esteganografía y para qué sirve?	6
1.2.1	Definiciones	6
1.2.2	Usos	6
1.2.3	Precauciones que deberían tomarse al usar la esteganografía	8
1.3	Futuro de la esteganografía	9
2	Técnicas esteganográficas básicas	10
2.1	En textos	11
2.2	En información digitalizada	12
2.2.1	Muestreo y cuantificación	13
2.2.2	Esteganografía en imágenes	13
2.2.3	Esteganografía en audio	15
2.2.4	En lugares todavía más extraños	16
2.3	En dispositivos	17
2.4	En protocolos	18
2.5	Otras técnicas, y técnicas complementarias	19
2.5.1	<i>Appending</i>	19
2.5.2	Introducción de “basura”	20
2.5.3	Eliminación de redundancias y patrones fijos	20
3	Estándares esteganográficos	20
3.1	No hay ninguno	20
3.2	Por qué?	20
3.3	Patentes	21
4	Aplicaciones de la esteganografía	21
4.1	Enviar mensajes cifrados donde están prohibidos	21
4.2	Digital Watermarking: prácticamente, la única aplicación comercial	22
4.3	Simple pasatiempos (?)	23
4.4	Nuestras sugerencias, y ejemplos ficticios	23
4.4.1	Rizar el rizo	23
4.4.2	Mensajes secretos a través de web, ftp, e-mail o news.	23
4.4.3	Johny Mnemonic	24
5	Implementaciones reales (software)	24
5.1	Software para Unix	25
5.1.1	Jsteg (patch para jpeg-v4)	25
5.1.2	Stext 1.0	25
5.1.3	StegParty 0.2	26
5.1.4	StegHide 0.3	26
5.1.5	OutGuess 0.13b	27
5.1.6	StirMark 1.0	28
5.1.7	GifShuffle, Snow y el cifrador ICE	28
5.1.8	MP3stego 1.1.15	30
5.1.9	GzipSteg (patch para gzip 1.2.4)	30
5.1.10	Covert_Tcp	31
5.2	Software para otras plataformas	34

5.2.1	Digital Picture Envelop 1.0 (BPCS-Steganography)	34
5.2.2	Otras aplicaciones	34
6	Apéndice: esteganografiar las herramientas esteganográficas	35
6.1	El problema	35
6.2	Posibles soluciones	35
7	Ejemplos para ilustrar este documento	36
8	Bibliografía	37

1 Introducción

Frases escogidas del Manifiesto Cypherpunk:

- “La privacidad no es secretismo. Una cuestión privada es algo que no queremos que todo el mundo sepa, pero una cuestión secreta es algo que no queremos que nadie sepa. La privacidad es la capacidad de revelarse selectivamente al mundo.”
- “La información no sólo quiere ser libre, anhela ser libre.”
- “La gente ha estado defendiendo su privacidad durante siglos mediante susurros, oscuridad, sobres, puertas cerradas, apretones de manos en clave y mensajeros. Las tecnologías del pasado no permitían una encriptación «fuerte», pero las actuales sí.”
- “Sabemos que el software no puede ser destruido y que un sistema ampliamente disperso no puede cerrarse.”
- “La criptografía va a extenderse en todo el mundo, y con ella los sistemas de transacciones anónimas que la hacen posible. Para que la privacidad se extienda tiene que formar parte de un contrato social. La gente tiene que unirse y usar estos sistemas para el bien común.”

Nosotros añadimos: el mundo real es tan complicado que tal vez la criptografía no sea suficiente para empezar. Afortunadamente, este problema tiene solución. No solo tenemos criptografía fuerte: también tenemos esteganografía.

1.1 Historia de la esteganografía

1.1.1 Los orígenes

La esteganografía es un conjunto de técnicas que permiten ocultar información dentro de otra, logrando así que informaciones confidenciales puedan pasar desapercibidas. El término proviene del griego *stegos*, cubierta, con lo que podríamos traducir esteganografía como "escritura tapada" o "escritura encubierta".

Su vínculo con el mundo griego puede ir mucho más allá del simple nombre. Aunque la criptografía y la esteganografía son conocidas sobre todo en su forma moderna, es decir, en el ámbito de la informática, su origen es tan antiguo como la existencia de la primera información que hubiera que mantener oculta a ojos ajenos. Es bastante probable pues que nacieran en el seno de las primeras civilizaciones, y en el ámbito militar. Se hace referencia a la esteganografía como "seguridad de transmisión", *TRANSEC* en la literatura militar.

El fin de ésta consiste en ocultar mensajes dentro de otros "inofensivos" de tal manera que una tercera persona o enemigo no pueda reconocer que el mensaje secreto está presente. Ésta es al diferencia con la criptografía donde esa tercera persona (enemigo) puede detectar, interceptar y modificar los mensajes. De todas maneras, incluso haciendo esto, el enemigo no puede violar los permisos de seguridad garantizados por un criptosistema.

Hay muchas maneras de representar mensajes simples ya sea, por ejemplo, mediante signos o dibujos. De todos modos la escritura no comenzó hasta que hubo un acuerdo sobre los símbolos, hasta que la reproducción de éstos fuera fácilmente comprensible. Se cree que este proceso comenzó en la antigua Mesopotamia (actual Irak) alrededor del 3.000 a.c.. En esta época, los sumerios vivían en el sur y los acadios en el norte. Tenían un acuerdo y comerciaban entre ellos por muy diferentes que fueran sus lenguas, así que las primeras tablillas (escritura cuneiforme) recogían básicamente transacciones agrícolas. El gran salto vino con la auténtica escritura cuyos signos empezaron a representar los sonidos del lenguaje hablado, cuando los pictogramas no se usaban para representar el objeto en sí. Esta forma de escritura se volvió extremadamente elaborada; GEORGE JEAN, profesor de lingüística de la universidad de Maine (Francia) escribió: "Era un arte confinado a aquellos que sabían cómo inscribir signos y a quien entendía los diferentes significados que debían tener dependiendo del contexto. En Babilonia y Asiria, hasta cierto punto, los escribas constituían una casta aparte, a veces, quizás, se volvían más poderosos que los cortesanos analfabetos o incluso que los mismos soberanos.

1.1.2 El Manuscrito Voynich, un misterio sin resolver.

En 1912, Wilfrid M.Voynich, anticuario norteamericano, compró un manuscrito medieval de 235 páginas en lo que parecía ser un lenguaje desconocido con singularidades, figuras humanas, dibujos astrológicos. Las personas y costumbres parecen, en general, europeas. La escritura pudo haber sido desarrollada a partir de los antiguos números arábigos y abreviaciones del latín medieval. Nadie lo sabe, pero la cantidad de ilustraciones sugieren que es un libro de alquimia, que alguien ha querido mantener en secreto.

El manuscrito tiene varias partes identificadas por las ilustraciones (aunque no hay garantía de que estas sean el tema de las secciones): una sección herbal (la mayoría no identificadas y plantas fantásticas), una sección astronómica (con la mayoría de signos del zodiaco), una sección biológica (con dibujos de anatomía), una sección cosmológica (con círculos, esferas celestes y estrellas), una sección farmacéutica (con recipientes y partes de plantas) y una sección de recetas.

Además hay varias secuencias clave a lo largo del libro, antigua escritura alemana (probablemente añadida después), en la sección astrológica, los nombres de los meses (probablemente añadido más tarde), varios ejemplos de extrañas escrituras (diferentes del resto del manuscrito), texto en escritura no Voynich donde se lee en el último folio algo como "michiton oladabas..." sugiriendo una clave para la descifración.

Este manuscrito parece, un poco, como un montón de cosas, pero realmente como nada. Descifrar este código ha sido, hasta el momento, una ardua e inútil tarea. El análisis computacional lo único que ha hecho ha sido incrementar la confusión. Los códigos de principios del siglo XVI derivaban de "The stenographica" de JOHANNES TRETHEMIUS, obispo de *Sponheim*. Trethemius era un alquimista que escribió a cerca de la transformación de textos para ocultar mensajes. Tenía un número determinado de métodos limitados al uso militar, alquimista, religioso o político hasta bien entrado el siglo XVII. Todavía no parece que el *Manuscrito Voynich* tenga alguna relación con los códigos legados por Trethemius. Hoy el manuscrito se encuentra en la universidad de *Yale*, en la "*Beinecke Rare Book Library*".

Este es un gráfico generado por ordenador de una muestra del folio10:

Currier-D'Imperio transcription

Friedman (FSG) transcription

<f10r> {D'Imperio A}

```
<f10r.1> BSOQ9.ZOR.OQ089.SORS9.BS080E.S0BS0E.9BSFOJ-
Facoñc g. áo? .oñc oðg. ao?acg. Facoðo s. acofaco s. gñall oñf-
<f10r.2> 8SC9.QOOR.SAR.SP9.OCSAT.OP9PSOE.OF9.8AM.CP98-
ðaccg. ðñc o o? . aar. allg. ocaar? . oñgñlaco s. oñg. ðawñ. cñggð-
<f10r.3> 4OPOR.OP99.8AM.S0Q9.4OP99.SOE.OR.9P9.89.89-
4oñl o? . oñlaccg. ðawñ. acoñc g. 4oñlaccg. aco s. o? . gñgg. ðg. ðg-
<f10r.4> 2OR.SAM.SQ9.QOX9.OR.AM.SP9OR.8OU.089-
2o? . aawñ. añc g. ðñc oñc g. o? . awñ. alllaco? . ðowñ. oðg-
<f10r.5> 40FS9.40PSOE.SOE.Q9#
4oñlaccg. 4oñlaco s. aco s. ðñc g#
<f10r.6> 9SCOR.Q9.SOR.QAM.40Q0E9.89.S9.PAM.Z9-
gacc o? . ðñc g. aco? . ðñc awñ. 4oñc o s g. ðg. accg. llawñ. ág-
<f10r.7> 8S9.40FSOE.9FSAM.9F9.8AM.Q.8AM.8AT.AJ-
ðaccg. 4oñlaco s. gñllawñ. gñgg. ðawñ. ðñc. ðawñ. ðawñ. añf-
<f10r.8> 40PSOR.SOR.OPOE.SOE.SOEOR.SOE.8AM.8AR-
4oñlaco? . aco? . oñl o s. aco s. aco s o? . aco s. ðawñ. ða? -
<f10r.9> 09FSOR.ZOR.SOR.S9.FA3.89.S08AM-
o gñlaco? . áo? . aco? . accg. llawñ. ðg. acoðawñ-
<f10r.10> 04OPOR.OPOR.W9.QOR.O2AN.9POM-
o4oñl o? . oñl o? . ðñc g. ðñc o? . o2awñ. gñllawñ-
<f10r.11> R0PSOZOR.40P9.4OPOR.Q98.OPAR-
2oñlaco? . 4oñl g. 4oñl o? . ðñc gð. oñl a? -
<f10r.12> R08AM.8AM.40PS9.4OPOR#
2oðawñ. ðawñ. 4oñlaccg. 4oñl o? #
```

```
# page 19
# folio 10r
# line ends added from Currier's transcription
# Kraus XXX (bottom)
# Currier's language A, hand 1
# herbal
```

```
PTOZHG.SOR.OHZ08G.TORTG.PT080ETOPTAE.GRT.DOK-
Facoñc g. áo? .oñc oðg. ao?acg. Facoðo s acofaco s. gñall oñf-
8TCG.HZ0OR.TAR.THG.O2.TAIR.OHG.HTOE.ODG.8AM.CHG8-
ðaccg. ðñc o o? . aar. allg. o? . aar. oñg. llaco s. oñg. ðawñ. cñggð-
40HOR.OHTG.8AM.TOHZG.4OHTG.TOE.OR.GHG.8G.8G-
4oñl o? . oñlaccg. ðawñ. acoñc g. 4oñlaccg. aco s. o? . gñgg. ðg. ðg-
2OR.TAM.THZG.HZO.DZG.OR.AM.THTOR.801IR.08G-
2o? . aawñ. añc g. ðñc. ðñc g. o? . awñ. alllaco? . ðoñl o? . oðg-
40DTG.4OHTOE.TOE.HZG=
4oñlaccg. 4oñlaco s. aco s. ðñc g=
GTCOR.HZG.TOR.HZAM.40HZ0EG.8G.TG.HAM.SG-
gacc o? . ðñc g. aco? . ðñc awñ. 4oñc o s g. ðg. accg. llawñ. ág-
8TG.40DTOE.GDTAM.GHG.8AM.HZ8AM.8AIR.AK-
ðaccg. 4oñlaco s. gñllawñ. gñgg. ðawñ. ðñc. ðawñ. ðawñ. añf-
4OHTOR.TOR.OHOE.TOE.TOEOR.TOE.8AM.8AR-
4oñlaco? . aco? . oñl o s. aco s. aco s o? . aco s. ðawñ. ða? -
0GDTOR.SOR.TOR.TG.DAIM.8G.T08AM-
o gñlaco? . áo? . aco? . accg. llawñ. ðg. acoðawñ-
04OHAR.OHOR.FZG.HZOR.O2AN.GHOM-
o4oñl a? . oñl o? . ðñc g. ðñc o? . o2awñ. gñllawñ-
ROHTO.SOR.4OHG.4OHOR.HZG8.OHAR-
2oñlaco? . áo? . 4oñl g. 4oñl o? . ðñc gð. oñl a? -
08AM.8AM.4OHTG.4OHOR=
oðawñ. ðawñ. 4oñlaccg. 4oñl o? =
```

by G.Landini@bham.ac.uk

La universidad de Yale te hace firmar un acuerdo en el que te comprometes a no publicar ninguna imagen o datos que en él aparecen. Aunque últimamente ha liberalizado su política en

cuanto al uso de imágenes del manuscrito Voynich. TAKESHI TAKAHASHI, “*a list member*”, obtuvo permiso para transcribir el manuscrito por completo y emplear los resultados a su antojo.

Podríamos extendernos todo lo que quisiéramos acerca de este misterioso manuscrito pero no es el objeto de este trabajo, a no ser que lográsemos descifrarlo. Únicamente pensamos que sería interesante introducir el trabajo con lo que algunos llaman "el manuscrito más misterioso del mundo".

1.1.3 Esteganografía: desde el origen de la humanidad hasta nuestros días

Aunque por supuesto no podemos hablar de quién inventó la criptografía o la esteganografía, y quizás verdaderamente nadie pueda demostrarlo, hemos encontrado casualmente uno de los primeros usos de la esteganografía que fueron documentados. El libro es *Historias*, de HERÓDOTO, en concreto el *Libro V*. La situación: la rebelión de las ciudades jonias de Asia Menor. ARISTÁGORAS, tirano de Mileto, es instado por el rey de Persia a organizar una expedición contra la isla griega de Naxos. Pero el también jonio HISTIEO, sabiendo que ARISTÁGORAS duda de permanecer leal a Persia, quiere ponerse en contacto con él para pedirle que se una a la rebelión; citamos textualmente, en una traducción no muy clara pero suficiente:

"... Aristágoras no pudo cumplir la promesa a Artafrenes. A la vez le abrumaba el gasto de la expedición que se le reclamaba, y tenía miedo porque el ejército había fracasado y por haber incurrido en la cólera de Megabates, y pensaba que se le arrebataría el poder soberano de Mileto. Como temía todas y cada una de estas cosas, proyectó una rebelión: pues coincidió también que el "tatuado" en la cabeza llegó de Susa de parte de Histieo, indicándole a Aristágoras que hiciera defección del rey. En efecto, Histieo, al querer indicar a Aristágoras que defecionara, no pudo indicárselo de ninguna otra manera segura por estar vigilados los caminos; él, al más leal de sus esclavos, tras repararle la cabeza, lo tatuó y esperó a que de nuevo le crecieran los cabellos y, tan pronto como le hubieron de nuevo crecido, lo envió a Mileto, no encomendándole ninguna otra cosa sino que, cuando llegara a Mileto, invitara a Aristágoras a que, tras repararle los cabellos, mirara en su cabeza. Y los tatuajes querían decir rebelión..."

Simple, pero eficaz. Aristágoras recibió el mensaje, y las ciudades jonias se alzaron contra el dominio persa. Afortunadamente, los puestos de vigilancia persas no tenían sistemas de detección de tráfico de información esteganografiada (es decir, tijeras en este caso :-)

La esteganografía ha sido durante mucho tiempo el método de comunicación favorito de líderes militares y políticos. HERÓTODO narró una historia acerca de un noble de Medea que escondió un mensaje para un posible aliado en el vientre de una liebre despellejada que fue entregada por un mensajero disfrazado de cazador. Desde el siglo primero, *Anno Domini*, hasta la Segunda Guerra Mundial, los espías utilizaban leche, vinagre y zumo de frutas como tintas invisibles. Con estas tintas, lo único que hay que hacer para revelar el mensaje secreto es calentar la hoja. Otras tintas más sofisticadas requieren que el receptor aplique una sustancia química.

El espía nazi GEORGE DASH escribió mensajes en su pañuelo con una solución de sulfato de cobre, que permanecían invisibles hasta que se exponían a vapores de amoníaco. Otra técnica desarrollada por los nazis durante la II Guerra Mundial fue llamada *microdot*, "micropunto". J. EDGAR HOOVER (director del FBI 1924-1972) la llamó la "obra maestra del espionaje enemigo". Un microdot era una fotografía diminuta, del tamaño de un punto. Cuando se revelaba podía reproducir una página mecanografiada con una claridad asombrosa. Era extremadamente difícil de detectar y los alemanes los utilizaban para transmitir grandes cantidades de datos impresos y planos.

Durante esta guerra, en la sala 40 del Almirantazgo en Londres, el *Manuscrito Voynich* pasó por las manos de casi todos los criptoanalistas del mundo. Los marines americanos destacados en el Pacífico durante la II Guerra Mundial no eran menos expertos en esteganografía. Reclamaron a los indios Navajos, denominados "*codetalkers*", para "encriptar" los mensajes secretos transmitidos vía radio. Usaban su lengua nativa que es extraordinariamente difícil y extremadamente ininteligible.

Estimaron en 28 las personas no navajas que pudieran hablar y entender esta lengua y ninguna de ellas era alemana o japonesa. Por si eso no fuera suficiente los "codetalkers" utilizaban una jerga. De esta manera, incluso los conocedores de esta lengua no podrían entenderles si la jerga no les resultaba familiar.

Durante la Guerra del Golfo, algunos navajos en EEUU quisieron enviar felicitaciones a sus seres queridos; al utilizar su lengua salvaron la censura de la Radio de las Fuerzas Armadas.

1.2 Qué es realmente la esteganografía y para qué sirve?

1.2.1 Definiciones

En casi todos los lugares donde hemos encontrado información sobre esteganografía hemos visto, antes que nada, una definición más o menos buena de lo que es la esteganografía. Seguramente esto pasa porque es algo bastante desconocido, a pesar de su antigüedad, y todos los autores creen que antes de empezar a hablar sobre ello conviene definirlo. Para unos es un arte, mientras que para otros es una ciencia; y la verdad es que a veces resulta difícil distinguir entre ambas cosas porque la esteganografía no solamente requiere conocimientos matemáticos y técnicos bastante avanzados sino que además requiere una buena dosis de imaginación y originalidad. Al final nos gustó la idea de MARKUS KUHN, que dice que es un arte y una ciencia. Como llamarlo "arte" es a la vez más práctico y más literario, usaremos aquí ese nombre.

¿Y en qué consiste ese arte? Como hemos dicho, hay multitud de definiciones. Algunos dicen simplemente que es el arte de comunicarse ocultando la propia comunicación. Sin embargo hay definiciones más amplias que no limitan la esteganografía a la comunicación y dicen que la esteganografía es el arte de ocultar información dentro de otra información; a esta definición habría que añadir que la información ocultada normalmente es sensible mientras que la otra es totalmente inofensiva. La forma más clara de explicar lo que es la esteganografía parece que es la que empieza comparandola con la criptografía: si tenemos que proteger la información secreta (almacenada o en transmisión) de un "enemigo" existen dos aproximaciones básicas. La primera aproximación es la criptografía, que de alguna manera codifica la información para protegerla de accesos no autorizados con diversas técnicas, asegurando que un atacante externo no podrá leerla (privacidad), modificarla/corromperla (integridad), etc. Pero el atacante normalmente sabrá que esa información codificada existe e incluso podrá acceder a ella. La segunda aproximación es la esteganografía, que oculta la propia existencia de la información secreta en otra información. De hecho sería más correcto decir "en otros lugares" más que "en otra información" porque hay técnicas (que veremos a lo largo de este documento) muy originales que consiguen poner la información en lugares totalmente insospechados, y no solo aprovechan medios lógicos (estructuras de datos, software) para ocultar la información sino que aprovechan características físicas de ciertos dispositivos para conseguir sus propósitos. Además, debemos considerar también las técnicas históricas que no utilizan para nada los medios técnicos actuales.

1.2.2 Usos

Antes de empezar, citemos un curioso texto que encontramos en alguna rincón de la web. Estaba en inglés americano, pero lo hemos traducido y como es corto vale la pena citarlo íntegramente:

Existen tres formas básicas de ataque para obtener nuestra información encriptada.

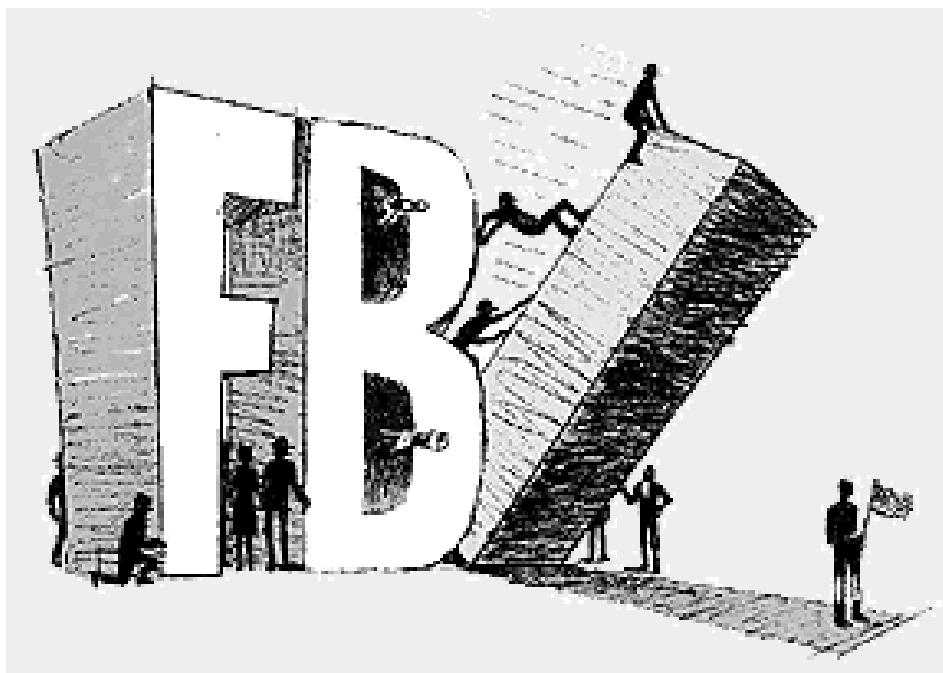
1) Adivinar el password. Por eso debemos escoger una frase difícil de adivinar antes que un simple password. Bajo la quinta enmienda de la constitución de los EEUU, no puedes ser obligado a testificar contra ti mismo. Por lo que tampoco lo estás de revelar tu password.

2) Un segundo ataque es el criptoanálisis. Este método depende de la calidad de la encriptación que utilizemos. De ella dependerá que tarden más o menos tiempo en crackearla o bien resulte inútil ya que el tiempo necesario para hacerlo exceda lo razonable.

*3) Al tercer ataque se le ha llamado **rubber hose** (traducible por "porra de goma"). Si nuestro enemigo es una entidad poderosa, como el Gobierno, no emplearán las dos primeras técnicas. En*

su lugar ellos vendrán tras nosotros golpeándonos con la porra hasta que revelemos el password o la información del fichero.

La encriptación te protege de los dos primeros ataques, pero no del tercero. Si el enemigo encuentra información encriptada en nuestro disco duro puede proceder al tercer método. Si la información encriptada se encuentra en un GIF seguramente no la encontrará y si la encuentra podemos decir de forma creíble que no sabemos nada acerca de eso (*plausible deniability*). Con suerte no se llevarán la información que escondemos.



Por supuesto el campo de aplicaciones de la criptografía y de la esteganografía es muy amplio. Es evidente que la criptografía por si misma resulta útil siempre que el mero hecho de que exista un secreto no deba permanecer oculto. Si se utilizan protocolos criptográficos bien diseñados que a su vez usan algoritmos criptográficos robustos (hay que recordar siempre que un atacante intentará romper la cadena por el eslabón más débil) la verdad es que un atacante no logrará obtener demasiada información de los datos secretos. Ya que esta información suele estar protegida mediante alguna clave, y el atacante lo tiene todo (los datos cifrados, el algoritmo, todos los detalles del protocolo, etc) excepto esa clave, lo único que descubrirá en un tiempo razonable es que existe información a la que él no puede acceder, y tal vez se hará una idea de las dimensiones de los datos protegidos, pero nada más. Por lo demás, si se usan buenos algoritmos la información cifrada parecerá totalmente aleatoria y no dará ninguna información del texto en claro.

Pero sucede que en algunos casos la criptografía no es suficiente. A veces la mera protección criptográfica de los datos no cumple las especificaciones de un problema de seguridad concreto. Cifrar los datos hace que un atacante no tenga acceso directo a los mismos sin la clave, pero no hay que olvidar que esos datos se encuentran en un mundo real bastante complicado, y en algunas situaciones no podemos tolerar ni siquiera que el “enemigo” sospeche que tenemos algo que ocultar. En el documento de ANDERSON, NEEDHAM Y SHAMIR sobre su sistema esteganográfico de ficheros nos hemos encontrado con numerosos argumentos en favor de la esteganografía, con unos ejemplos muy gráficos: situaciones de guerra, dictaduras, secuestros, atracos, abusos de poder, etc, en las que no solo importa codificar la información secreta de forma que solo quien tiene la clave pueda leerla, sino que es importante ocultar la propia existencia de esa información. En realidad, ese documento se refiere a una propiedad más avanzada y muy importante que puede obtenerse a través de la esteganografía: la llamada “*plausible deniability*”, que vendría a ser lo opuesto al no-rechazo de origen que puede obtenerse mediante protocolos de correo electrónico certificado

usando algoritmos de cifrado y de hash.

Después de todo lo dicho, puede parecer que la esteganografía por sí sola carece de utilidad, puesto que se basa en que el atacante ignora la existencia de la información, pero si sabe (o sospecha) que ésta existe y conoce el método usado para esteganografiarla puede extraerla sin demasiadas dificultades. Veremos que de hecho ese uso de la esteganografía sin más puede resultar útil sobre todo para hacer “marcas de agua” (watermarking) digitales sobre imágenes y sonido para evitar la violación de copyrights, etc; pero la verdad es que la esteganografía se vuelve realmente útil cuando es usada como complemento de la criptografía. Veremos con más detalle los motivos técnicos de esto en el apartado de *técnicas esteganográficas*, pero la idea es que si la información esteganografiada ha sido previamente comprimida y encriptada su aspecto es prácticamente aleatorio, y solo descriptandola con la clave adecuada y luego descomprimiendola recobra su sentido. Haciendo las cosas de esta manera si un atacante sospecha que en algún lugar hay información esteganografiada y además conoce el método usado para esconderla allí, al extraerla se encontrará con una secuencia de bits de aspecto totalmente aleatorio y no tendrá forma de saber si sus sospechas eran ciertas o por el contrario no hay información esteganografiada y solamente ha extraído ruido. Ni siquiera sabrá si el método utilizado era el correcto, tendrá que probar varios (¡y hay muchos!).

En cualquier caso la esteganografía puede usarse para engañar a muchos tipos de “enemigos”. Veremos que hay técnicas esteganográficas que pueden burlar fácilmente a un observador humano, pero son fácilmente detectables algorítmicamente, como ciertas técnicas de ocultación en audio; otras técnicas son muy evidentes a simple vista pero algorítmicamente son casi imposibles de detectar, como la generación de textos semanticamente pobres (pero sintacticamente correctos y estadisticamente normales) en función del mensaje secreto; y la verdad es que unas pocas aplicaciones consiguen engañar a casi todos los humanos y además algorítmicamente son muy difícilmente detectables. También veremos en este documento como la gran variedad de aplicaciones esteganográficas existentes, la naturaleza originalmente retorcida de la mayoría de ellas, y la total ausencia de estándares en la práctica, contribuye a crear confusión en el “enemigo” y dificulta su labor.

1.2.3 Precauciones que deberian tomarse al usar la esteganografía

Varios programas comerciales y gratuitos permiten hacer esteganografía, ya sea por sí mismos o formando parte de un paquete completo de seguridad en las comunicaciones. Su justificación es la siguiente: si Patricia quiere enviar a Guillem un correo electrónico de forma segura, puede utilizar alguno de los muchos programas de encriptación de e-mail. A pesar de ello, un fisgón (suponiendo que no tenemos enemigos) podría interceptar el mensaje y, aun siendo incapaz de leerlo, sabría que Patricia está enviando a Guillem un mensaje secreto. La esteganografía permite a P. comunicarse con G. de forma secreta. Ella puede coger su mensaje y esconderlo en un fichero tipo GIF con la imagen de dos jirafas.

Cuando el fisgón intercepte el mensaje, todo lo que verá será una fotografía de dos jirafas; no tendrá ni idea de que se está enviando a G. un mensaje secreto. Incluso ella podría encriptar el mensaje antes de esconderlo, como protección adicional. A partir de ese momento, la comunicación esta protegida con la fuerza de la criptografía y la esteganografía. Hasta aquí, muy bien; además no hemos dicho nada excesivamente nuevo. Pero no es así como funciona realmente en la práctica. El fisgón no es estúpido. Tan pronto como vea la fotografía de las jirafas, comenzará a sospechar. ¿Porqué enviará P. a G. una fotografía de dos jirafas?. ¿Acaso G. colecciona jirafas?. ¿Es un artista gráfico?. Han estado P. y G. enviándose el uno al otro esa misma imagen de las jirafas durante semanas. ¿Acaso mencionan siquiera esa fotografía? ¡Conviene actuar con un poco de sentido común!

El objeto de la esteganografía es esconder la existencia de un mensaje, ocultar el hecho de que las partes se están enviando algo distinto a unas inocuas fotografías. Esto sólo funciona cuando se utilizan los patrones de comunicación usuales. Si P. y G. ya intercambiaban, de forma regular, ficheros adecuados para ocultar mensajes esteganográficos, el fisgón no podrá saber qué envíos contienen los mensajes, si es que los hay. Si P. y G. cambian sus patrones de comunicación

para esconder mensajes, no funcionará. Un fisgón sospechará. Se deben tomar una serie de precauciones. No se debe utilizar la imagen de ejemplo que viene con el programa; el fisgón lo reconocería enseguida. No utilizar la misma imagen una y otra vez; el fisgón buscará las diferencias entre los envíos que revelen la existencia de un mensaje secreto. No utilizar una imagen que se haya descargado de la red; el fisgón puede fácilmente comparar la imagen que se está enviando con la imagen de referencia (debemos suponer que él monitorizó su descarga o que ha buscado y encontrado en la red esa misma imagen). También deberíamos tener una buena excusa para explicar porqué estamos enviando imágenes una y otra vez. Y esa excusa debería existir antes de que comenzase a enviar mensajes esteganográficos, y también después. De otra forma no habremos ganado realmente nada, ya que se podría sospechar de nuestro cambio de comportamiento.

La esteganografía también puede utilizarse para ocultar ficheros en el disco duro. Esto también es algo problemático. Supongamos que la policía secreta nos arresta (por error, ¡por supuesto!) y comienza a buscar en nuestro disco duro. Tenemos un montón de fotos en él. Pero además tenemos el programa esteganográfico en el disco duro, por tanto la policía secreta sospechará (trataremos este tema más a fondo en el apéndice). Ellos podrían tratar de descargar las mismas fotografías de la red y buscar las diferencias que revelasen la existencia de un mensaje oculto. O pueden simplemente suponer que tenemos mensajes ocultos en alguna parte.

En cualquier caso, no hay seguridad absoluta ni en el campo de la criptografía ni en el de la esteganografía (ni en ningún otro campo). Los atacantes experimentados pueden encontrar agujeros en sistemas tanto criptográficos como esteganográficos. El objetivo es hacer que esto sea lo más difícil posible. Es práctica habitual usar una clave para manejar un generador de números pseudoaleatorios y elegir un subconjunto de los pixels para ocultar el mensaje. Estamos seguros de que hay ataques estadísticos contra esto que pueden llegar a averiguar el generador de números pseudoaleatorios y otros que no, pero también sostenemos que eso no es algo que se pueda llevar a cabo en un día o dos. Con unas pocas precauciones razonables, un mensaje puede ser ocultado bastante bien. Hay muchas cámaras digitales que cuestan muy poco dinero. ¡Es fácil generar nuevo contenido en gran cantidad!. Mucha gente envía fotos y más fotos. Mucha gente envía ficheros de voz con sus mensajes. Mucha gente envía dibujos de los niños...

1.3 Futuro de la esteganografía

Veremos a lo largo de este documento que la esteganografía tiene muchas aplicaciones, pero como es algo basado pura y simplemente en el secreto, la aleatoriedad y la improvisación a la hora de esconder (y guardar o transmitir) una información, difícilmente se convertirá en algo popular y de uso extendido porque en general no es necesario tanto secreto. Esto la diferencia de la criptografía, que hoy en día es de uso general y está presente en todas partes (teléfonos móviles, comercio electrónico, etc) aunque la mayoría de la gente no se da cuenta.

Aunque existen varias patentes sobre técnicas esteganográficas, veremos que no hay estándares reales y es de esperar que eso siga así porque desde el momento en que se establece un estándar esteganográfico y alguien anuncia que lo adopta, todo deja de tener sentido: ¡eso equivale a confesar que hay secretos que ocultar y explicar como se ocultan, y ya no es necesaria la esteganografía!

Sin embargo la esteganografía siempre ha existido, es casi tan antigua como la escritura (o incluso más) y no parece que vaya a desaparecer. Por una parte tenemos aplicaciones como el *watermarking* que es prácticamente la única aplicación comercial que tiene y que resulta bastante útil y empieza a extenderse. Por otra parte, tenemos la esteganografía de toda la vida explicada en el apartado de *historia* que sigue perfectamente vigente. Y además tenemos una increíble cantidad de aplicaciones esteganográficas (la mayoría *freeware* o *shareware*) creadas por gente de todo tipo: desde los que lo hacen por simple diversión hasta los que lo hacen con orientaciones de seguridad. No hay que olvidar que en algunos países como Francia o China el uso de la criptografía es algo ilegal, peligroso en Francia y muy peligroso en China; y que en algunos países como el nuestro o como USA “es posible” que los gobiernos estén escuchando. Y seguro que las agencias de seguridad de todos los gobiernos usan sus propios métodos esteganográficos cuando les conviene.

En resumen: la esteganografía tiene futuro, está destinada a seguir (como siempre) existiendo en la sombra.

2 Técnicas esteganográficas básicas

Sobre esteganografía hay poca documentación didáctica. Pueden encontrarse unas pocas referencias históricas, unos cuantos documentos bastante teóricos escritos por autoridades en matemáticas y criptografía (SHAMIR, etc), y la escasa documentación de unos pocos programas (no demasiado comerciales) hechos en sus ratos libres mayoritariamente por técnicos en informática y comunicaciones.

Trataremos algunas implementaciones concretas y sus retorcidas técnicas en apartados posteriores, en este apartado estudiaremos brevemente las técnicas esteganográficas más elementales simplemente para mostrar que la esteganografía es algo que puede llevarse a la práctica con bastante éxito.

Podemos clasificar las técnicas esteganográficas de muchas maneras. Si definimos “información portadora” como la información aparentemente inocente en la cual esteganografiamos la información secreta, la clasificación más evidente viene dada por la naturaleza de la información “portadora” del secreto, que es la que seguiremos en este apartado porque en general da lugar a técnicas bastante diferentes. Es interesante analizar antes otras clasificaciones según otros aspectos de la esteganografía.

Las técnicas esteganográficas generalmente se basan en modificar muy ligeramente las partes más superfluas y/o aleatorias de la información portadora para esconder la información secreta de la forma más disimulada posible, pero existe otra aproximación que consiste en generar una portadora específica para esconder esa información. Por ejemplo, podemos esconder información en una fotografía digitalizada ya existente introduciendo modificaciones leves pero muy estudiadas en la información gráfica, o podemos generar una nueva imagen diseñada específicamente para que sea aparentemente inocente pero que en el fondo contenga el mensaje secreto codificado de alguna manera. Normalmente modificar una portadora existente produce una ligera distorsión que puede ser detectable algorítmicamente (haciendo análisis frecuenciales de las imágenes o los sonidos, etc), mientras que generar una portadora específicamente diseñada para contener datos esteganografiados tiende a producir resultados más fácilmente detectables por observadores humanos. Es evidente que si usamos siempre un algoritmo diseñado para esteganografiar información en fractales de Mandelbrot de forma que sea estadísticamente imposible encontrar esa información, un programa espía que analice el tráfico de una red no encontrará nada extraño, porque seguramente utilizará criterios estadísticos... pero puede que un espía humano piense que mandar dos o tres gráficos de ese tipo cada día resulta sospechoso.

En general la esteganografía se aprovecha de la naturaleza redundante o aleatoria de parte de la portadora para esconder la información secreta, aunque no siempre es así: a veces simplemente esconde la información en lugares inesperados. En cualquier caso, es muy conveniente que la información secreta que vamos a esteganografiar esté comprimida y encriptada porque así tiene una apariencia realmente aleatoria, casi parecer ruido blanco (se le llama así por analogía con la luz blanca, que tiene componentes de todas las frecuencias). De esta manera si un atacante consigue romper todo el esquema esteganográfico tendrá que romper además el esquema criptográfico para obtener la información y para estar seguro de que lo ha hecho bien, y eso no es nada fácil. Comprimir la información antes de encriptarla tiene dos efectos beneficiosos: por una parte, reduce el tamaño de la información; y por otra, elimina redundancias. La reducción de tamaño es más que deseable porque generalmente las portadoras aceptan poca información esteganografiada, y la eliminación de redundancias hace más difícil que un atacante pueda romper con éxito el esquema criptográfico porque muchos de los ataques existentes se basan precisamente en aprovechar redundancias en el mensaje.

Como complemento a este trabajo, tenemos el siguiente programa que calcula la “entropía” de un fichero. Hay programas mucho mejores, pero también mucho más largos, que calculan además muchos datos estadísticos (media, varianza, correlación, valor Montecarlo de PI, ratio máximo de compresión que podría alcanzarse, etc). Sin embargo como primera aproximación para analizar un fichero está bien conocer la entropía. Podemos extraer muchas conclusiones de ese dato, a menudo acertadas: si tiende a 8 bits por byte el fichero tiene un aspecto bastante aleatorio, probablemente será difícil comprimirlo y seguramente se trata de información ya comprimida o encriptada; en

cambio, si la entropía no tiende a 8 bits por byte es probable que el fichero puede comprimirse, y si ya ha sido comprimido y/o encriptado los algoritmos no eran demasiado buenos. Por otra parte, calculando la entropía antes y después nos sirve para ver (muy por encima) hasta que punto ha afectado la esteganografía a un fichero que hace de portadora.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define MAX    256
#define LOG2(X) log(X)/log(2.0)

int main (int argc, char **argv)
{
    FILE *in;
    int c;
    unsigned long int freqs[MAX], total;
    double p, h;

    fprintf(stderr,"Enthropy v0.1 by GCR 2000.02.07\n");
    if((argc!=2)||((in=fopen(argv[1], "rb"))==NULL)) {
        in=stdin;
        fprintf(stderr,"Reading from stdin\n");
    } else fprintf(stderr,"Reading from file %s\n",argv[1]);
    for(c=0,total=0;c<MAX;freqs[c++]=0);
    while((c=fgetc(in))!=EOF) {
        freqs[c]++;
        if((++total&0x3FF)==0) fprintf(stderr, ".");
    }
    for(c=0,h=0.0;c<MAX;c++) if((p=1.0*freqs[c]/total)!=0) h+=p*LOG2(1.0/p);
    if(total) printf("\nTotal %d bytes. H =%9.6f bits/byte.\n",total,h);
    fclose(in);
    exit(0);
}
```

2.1 En textos

El texto es un tipo de información muy concreto, con unas características muy peculiares que hacen que sea una buena portadora para la esteganografía. Para empezar, el texto es algo muy común, se trata de una forma muy extendida de guardar información y de comunicarse. Es algo que despierta muy pocas sospechas y que se usa para esteganografiar información desde hace muchísimo tiempo, mucho antes de la era digital. Además, el texto es altamente redundante, y eso se ve fácilmente en cuanto uno intenta comprimirlo. Y por si todo eso fuera poco, es un tipo de información que (precisamente gracias a su naturaleza redundante) normalmente tolera muchas modificaciones o incluso errores sin perder su significado original. Suele ir destinado a personas, y las personas son sistemas con una “heurística” bastante avanzada :-). Veamos pues varias formas de esteganografiar información en un texto. Podemos empezar por las técnicas que nada tienen que ver con la era digital y que pueden hacerse (por ejemplo) a mano, en una carta. Son las mejores para entender lo que es la esteganografía. Después seguiremos con técnicas que trabajan con textos digitales, en formato ASCII, etc.

Una forma de esteganografiar información en un texto consiste en utilizar la longitud de las palabras, o el número de palabras en cada línea, o el número de acentos o de apariciones de una letra concreta en cada línea, o en cada párrafo, para codificar la información. Por ejemplo, para contestar a una serie de preguntas de respuesta binaria (sí/no) podría establecerse un criterio como el siguiente: un número par de palabras en una línea significa “sí”; un número impar de palabras en cada línea significa “no”. Como otro ejemplo, podríamos utilizar el número de palabras en cada frase, o en cada párrafo, para codificar números; por ejemplo, números de teléfono. O podríamos utilizar leves variaciones en la escritura de ciertas letras para codificar la información: poner el punto ligeramente a la izquierda de la “i” significa que “no”, pero ponerlo ligeramente a la derecha de la “i” significa que “sí”. Existen miles de combinaciones posibles de este tipo, que

puede hacer cualquiera en un texto escrito, y eso es esteganografía porque esconde información (secreta) dentro de otra información (la portadora), siguiendo unas reglas que permiten separar de nuevo la información secreta de la portadora. Y como todas las técnicas esteganográficas, eso tiene implicaciones de seguridad muy importantes porque permite guardar e intercambiar mensajes secretos utilizando medios totalmente autorizados y sin despertar sospechas.

Sin embargo la esteganografía (como todo lo relativo a la información) expande sus posibilidades hasta límites insospechados con la llegada de la era digital. El simple hecho de utilizar una codificación como ASCII para tener textos en forma digital, con unos caracteres especiales para codificar los espacios, los finales de línea, etc, hace que los textos sean muy fácilmente manipulables y añade nuevas características explotables por la esteganografía.

Existen también (literalmente) miles de técnicas esteganográficas para texto digital. Veremos más adelante aplicaciones reales que esteganografían información en textos, pero en general podemos examinar algunas de las posibilidades que ofrece el texto en su forma digital. Por ejemplo, los espacios se codifican con un código ASCII más, un carácter igual que las letras con el código 0x20, solo que ese carácter tiene un espacio en blanco como representación gráfica. Pero existen otros caracteres que se imprimen en pantalla como un espacio en blanco, por ejemplo el carácter 0xFF. Ese carácter tiene el pequeño inconveniente de ser de 8 bits (ASCII puro es de 7 bits) pero en algunos contextos ese problema no es relevante; entonces, pueden utilizarse todos los espacios de un texto para codificar un bit en cada uno: si el “espacio” es un ASCII 0x20 leeremos un cero, mientras que si el “espacio” es un 0xFF leeremos un uno. Por supuesto esta técnica es fácilmente detectable mediante programa, y puede dar problemas en el correo electrónico (habrá que utilizar algún tipo de codificación para mandar mensajes con caracteres ASCII 0xFF) pero a simple vista no se nota. Otros programas codifican la información añadiendo un número variable de espacios al final de las líneas. Eso también es detectable, pero en general es algo que puede ocurrir en un texto normal de manera que puede ser difícil distinguir un texto normal de un texto con información esteganografiada.

Por supuesto existen técnicas mucho más sutiles: algunas frases o expresiones pueden escribirse de varias formas sin variar su significado, y podemos aprovechar eso para codificar información. Intuitivamente se ve que esta técnica permite esconder mucha menos información en un texto ya que para codificar unos pocos bits hay que encontrar una expresión que sea sustituible por otra sin variar el significado del texto, y además eso es lento porque requiere búsquedas en largos diccionarios, etc; pero como contrapartida, es prácticamente indetectable.

Otra técnica podría consistir en automatizar (con la ayuda de un programa) la utilización de la longitud de las palabras para codificar información. Al generar texto de esa forma, algorítmicamente y a partir de diccionarios y unas pocas reglas, el resultado sería estadísticamente correcto para un programa analizador de tráfico, pero un observador humano lo encontraría realmente sospechoso porque seguramente no tendría sentido.

En pocas palabras: la esteganografía en textos ofrece posibilidades casi ilimitadas, aunque las técnicas realmente seguras (casi indetectables) requieren grandes cantidades de texto portador para codificar cantidades razonables de texto secreto. Veremos ejemplos concretos más adelante.

2.2 En información digitalizada

La información digitalizada es el auténtico reino de la esteganografía de la era digital. Para ver por qué, hay que analizar un poco el proceso de digitalización en sí. En general, y en su forma más básica, consiste en transformar unos datos analógicos (con entropía “infinita”, con una cantidad de información “infinita”) en algo manejable, analizable por un sistema digital que solo trabaja con información binaria: unos y ceros. Esos datos analógicos pueden ser imágenes estáticas, imágenes en movimiento, sonidos... en el fondo cualquier magnitud física que podamos medir y nos interese registrar para su posterior análisis, procesado, o simple reproducción. Para convertirlos en información digital, manejable por un ordenador, hay que dar esencialmente dos pasos: muestreo y cuantificación.

2.2.1 Muestreo y cuantificación

El muestreo consiste en hacer lecturas de la magnitud que digitalizamos cada cierto intervalo de tiempo, y por tanto es la discretización del eje temporal, que de forma natural es continuo y consta de infinitos instantes. Existen teorías matemáticas (el TEOREMA DE SHANNON) que demuestran que si efectuamos el muestreo a una frecuencia mayor o igual al doble de la frecuencia máxima de la magnitud que queremos medir, no perdemos información (por eso si las frecuencias máximas que podemos oír los humanos rondan los 20Khz, la calidad CD-Audio se consigue muestreando a unos 44.1Khz, algo más del doble). Es decir, que en esas condiciones a partir de las muestras puede recuperarse la señal original. Por supuesto la señal original tiene componentes a todas las frecuencias porque como mínimo tiene superpuesto un ruido blanco gaussiano aditivo que no podemos evitar de ninguna manera, pero utilizando filtros adecuados (pasabajas) antes del muestreo se minimizan sus efectos. Para reconstruir la señal original hace falta un filtro a la salida; un matemático llamaría “interpolador” a ese filtro. Es decir, que (al menos teóricamente) en el muestreo no perdemos información. Entonces, la cantidad de información que tenemos sigue siendo infinita ya que los valores de la señal original en los instantes de muestreo siguen siendo continuos: por eso el siguiente paso es la cuantificación, que consiste en asignar un valor discreto (de entre todos los disponibles, en un rango finito) a cada una de las muestras. En este proceso si que perdemos información, puesto que teníamos información infinita, con valores continuos, y ahora tenemos algo finito y totalmente discreto. A modo de ejemplo, la calidad de CD-Audio asigna un valor de 16 bits a cada una de las muestras.

En suma, para grabar en calidad CD-Audio en estéreo (2 canales) si tenemos 44100 muestras de 16 bits por segundo tenemos un flujo total de datos de 1.4112Mbits/s, o 176.4Kbytes/s. O como otro ejemplo, si capturamos con una cámara digital una imagen estática de 1800x1200 pixels con muestras de 24 bits de color por pixel (1 byte por cada componente RGB) obtenemos unos 6.18Mbytes de información digitalizada.

Esta es la única forma de proceder, si lo que queremos son datos digitales no nos queda más remedio que perder información. Y no es tan grave puesto que en general las magnitudes que digitalizamos dan información visual o auditiva destinada a seres humanos, que tienen unos sentidos con unas limitaciones y unos umbrales mínimos más allá de los cuales no perciben diferencias. Los formatos de información digital de alta calidad superan ligeramente los umbrales humanos medios de manera que prácticamente todo el mundo percibe la información digitalizada sin notar que se ha perdido parte de la información original. En el ejemplo de la imagen fotográfica, el hecho de usar 24 bits por pixel implica que cada pixel puede tomar uno cualquiera de $2^{24} = 16777216$ colores diferentes... ¡pero el ojo humano más sensible ni siquiera distingue unos pocos millones! De la misma manera las imágenes en tonos de grises con muestras de 8 bits tienen disponibles 256 tonalidades de grises, que son muchas más de las que puede distinguir un ser humano normal.

De manera que en el proceso de digitalización se pierde información, pero como de todas formas nuestros sentidos (por otros motivos) no son capaces de captar toda la información existente lo que se hace es intentar que la información digitalizada sea más de la que pueden captar los sentidos de un ser humano medio para así dar una sensación realista a las imágenes y los sonidos digitalizados. *Eso es exactamente lo que hace que la información digitalizada sea tan adecuada para la esteganografía.*

2.2.2 Esteganografía en imágenes

Después de la (enorme) introducción a las técnicas esteganográficas en información digitalizada, la forma más básica de esteganografía en imágenes es casi evidente: si resulta que las muestras contienen más información de la que el ojo humano puede ver, podemos alterar ligeramente las muestras (llamaremos aquí muestras a los valores de color de cada pixel, ya que nos centraremos en imágenes digitalizadas porque las imágenes sintetizadas no son tan útiles para esteganografía, debido a su origen altamente determinista) sin que un observador normal lo perciba.

A modo de ejemplo, podemos centrarnos en las imágenes con 24 bits de color. Estas muestras tienen tres componentes de 8 bits que indican respectivamente los niveles (de 0 a 255) de los

componentes básicos del color: R (red, rojo), G (green, verde) y B (blue, azul). Con esto hemos visto que un pixel puede tomar $256 \cdot 256 \cdot 256 = 16777216$ colores diferentes, combinando diferentes tonalidades de esos colores básicos. Lo realmente interesante es que una pequeñísima variación en la tonalidad de un pixel practicamente no es detectable a simple vista. Por ejemplo, el color blanco puro tiene las tres componentes de RGB con valores máximos, a 255. Resulta que si lo comparamos a simple vista con un color gris muy cerca del blanco, como por ejemplo el que tiene las tres componentes RGB a 254, seguramente no podremos distinguirlos entre ellos o percibiremos una diferencia totalmente despreciable. Con un color cualquiera resulta que una pequeña diferencia en el valor de las muestras no es perceptible a simple vista. El ejemplo del gris claro y el blanco implica cambiar 3 bits de los 24 que tiene la muestra, y en algunos casos eso puede llegar a notarse. Pero empíricamente se ha comprobado que en general un cambio de un solo bit no se nota en absoluto.

Por supuesto los bits que pueden modificarse son los bits menos significativos de las muestras ya que provocan pequeños cambios en el valor de las mismas; un cambio en los bits altos provocaría grandes variaciones en la tonalidad que serían muy evidentes a simple vista. El número de bits que pueden modificarse en cada muestra varía según el caso (sobre todo según el tamaño de la muestra), pero como norma general no resulta demasiado bien modificar más de uno o dos bits por muestra.

Los ficheros gráficos con muestras de 8 ó 16 bits son menos adecuados para esteganografía porque en general tienen una estructura diferente. Dejando de lado las cabeceras y demás, los ficheros gráficos llamados TrueColor son los que tienen muestras de 24 bits (o más), y la información gráfica se guarda secuencialmente: los tres bytes RGB para cada pixel, uno detrás de otro, para todos los pixels de la imagen. Esta forma de guardar información se suele llamar “raw” (cruda) y aunque en ocasiones puede comprimirse en el fondo la estructura es esa. Los ficheros con muestras de 16 bits son normalmente más complejos porque los componentes básicos del color siguen siendo tres (RGB) pero tenemos 16 bits para guardarlos. Normalmente se usan 5 bits para dos componentes de color y 6 para el otro (existen variantes de 15 bits que usan 5 bits para cada componente), y los ficheros suelen contener también información “cruda” (comprimida o no) para cada pixel. Como las muestras son más pequeñas, y los colores posibles en las imágenes a 15 ó 16 bits (HiColor) son menos (32768 y 65536 respectivamente) las modificaciones pueden llegar a notarse, aunque sean de un solo bit. Y finalmente, las imágenes a color de 8 bits tienen una estructura totalmente diferente: permiten 256 colores diferentes, y tienen una paleta de $3 \cdot 256 = 768$ bytes que especifica para cada uno de los 256 colores de la imagen los tres componentes RGB utilizando 24 bits para cada uno. La información que viene a continuación es de un byte (8 bits) por pixel y son referencias a los índices de los diferentes colores de la paleta. Debido a esa estructura, es bastante más complicado hacer esteganografía en imágenes de 256 colores y las técnicas son algo diferentes; veremos algunas en el apartado de software real, pero básicamente consisten en hacer reordenaciones y permutaciones de la paleta, y las cantidades de información que pueden albergar estos ficheros es bastante limitada. Los ficheros de 256 grises son algo más fáciles de tratar porque el rango de color es continuo al igual que las tonalidades RGB de los ficheros de 24 bits y análogamente es posible alterar los bits más bajos (dentro de un orden) sin demasiados efectos visuales.

Por supuesto contamos todos estos detalles sobre los ficheros gráficos porque son precisamente los que permiten la esteganografía. La idea es que las imágenes digitalizadas a partir de imágenes reales (mediante escáner, cámara digital, etc) tienen los bits menos significativos de las muestras prácticamente aleatorios, puede decirse que son casi ruido. Esto, unido al formato utilizado para guardar esa información gráfica hace posible que modificar los bits bajos de las muestras no tenga prácticamente efectos visuales apreciables. Por tanto, podemos utilizar esos bits para esteganografiar la información: las imágenes pueden hacer de portadora. Podemos substituir los bits más bajos de las muestras por la información que queremos ocultar. Por eso es importante que la información que vamos a ocultar esté comprimida (y mejor si además está encriptada): así es estadísticamente indistinguible de los bits bajos originales.

Por otra parte, es algo conveniente y muy razonable elegir de forma pseudoaleatoria la secuencia de los pixels en que vamos alterando las muestras de color para esconder los bits de

información; es importante que sea de forma pseudoaleatoria porque es necesario (para recuperar la información) poder repetir la secuencia. Esto suele hacerse mediante generadores congruenciales de números pseudoaleatorios o en ocasiones utilizando funciones de hash realimentadas, o cifradores de flujo, siempre con la intención de dispersar uniformemente por toda la imagen la información esteganografiada y reducir así el impacto de la esteganografía complicando al mismo tiempo la labor de un posible atacante. Si se utiliza alguno de estos métodos es evidente que la “semilla” utilizada será imprescindible para la recuperación de la información y en cierto modo hará la función de “clave” del algoritmo esteganográfico. Si se combina la esteganografía con técnicas criptográficas puede ser una buena idea derivar de la clave proporcionada por el usuario no solo la clave de encriptación sino también esta semilla.

La explicación que hemos dado en este apartado sobre esteganografía en imágenes solo contempla el caso más sencillo y más implementado, que sirve para mostrar que la esteganografía es algo posible, pero existen muchísimas otras técnicas. Las imágenes en TrueColor son algo bastante habitual, pero guardar la información “cruda” es algo que ocupa muchísimo espacio aunque se utilicen técnicas de compresión del estilo de Huffman o variantes de Lempel-Ziv (llamadas *lossless* porque permiten en la descompresión recuperar la información binaria original, sin pérdidas de ningún tipo). Son formatos de imagen poco adecuados para la transmisión de datos, pero excelentes para fotografía de alta calidad, y algunos de los formatos más extendidos soportan ese tipo de imágenes, como Tiff, Targa, BMP, por citar algunos. Sin embargo, existe otra categoría de formatos gráficos mucho más complejos y mucho más avanzados que utilizan técnicas llamadas “*lossy*” para reducir drásticamente el tamaño de la imagen. Eso significa que se pierde información, pero son extremadamente eficientes (llegan a ratios de compresión de 50 a 1 o incluso más), dan una calidad aceptable (configurable por el usuario, más calidad implica ficheros más grandes) y en general son los más usados para fotografía digital: uno de ellos es el JPEG, sin duda el formato más usado de la Web. Veremos más adelante cómo puede esteganografiarse información en un JPEG, gracias a una aplicación real que analizaremos.

Para terminar este apartado, vamos a dar una idea aproximada de la cantidad de información que puede esteganografiarse en una imagen. Supongamos un caso en el que alteramos un solo bit de cada muestra, algo bastante seguro; si tenemos una imagen de 1800x1200 a 24 bits por pixel, capturada por una cámara digital, tenemos $1800 \cdot 1200 = 2160000$ muestras de 24 bits. Suponiendo que podemos guardar un bit de nuestro mensaje secreto en cada una de las muestras, alterando el bit bajo de uno solo de los tres componentes de color de la muestra (eso es bastante discreto) resulta que ese gráfico tiene capacidad para almacenar (¡como máximo!) 270000 bytes de información. Hay que tener en cuenta que eso es un máximo teórico y que depende de la imagen portadora que utilicemos (algunas imágenes toleran mejor la esteganografía que otras), pero considerando que se trata de información que podemos comprimir previamente, eso es mucha información... Tal vez varias imágenes comprimidas en JPEG podrían viajar dentro de un BMP de esas dimensiones. Y a parte de todo esto, existe una técnica de esteganografía “de alta capacidad” llamada *BPCS* diseñada por el japonés EIJI KAWAGUCHI que permite introducir cantidades mucho mayores de información en el mismo tipo de imágenes. La veremos en la parte de implementaciones reales.

2.2.3 Esteganografía en audio

Las señales de audio son todavía más simples que los gráficos, pero la esteganografía básica puede hacerse de forma muy similar. Sencillamente, las muestras de audio son más simples, ya que solo guardan un valor (la amplitud de una magnitud física) y no están formadas por tres componentes RGB como las muestras visuales; y además se toman siguiendo una sola dimensión, el eje temporal, y no las dos dimensiones de un plano como en el caso de las imágenes. Además, el órgano sensorial humano al que van destinados los sonidos es distinto: el oído humano, en lugar de los ojos.

El audio digitalizado puede guardarse en una gran variedad de formatos y calidades. Las muestras pueden tomarse a frecuencias que oscilan entre 8Khz y 44Khz que son aproximadamente “calidad telefónica” y “calidad CD”, respectivamente. Y el tamaño de las muestras es típicamente de 8 bits o de 16 bits. Eso, para cada canal de audio; estéreo implica dos canales de audio. Evidentemente para entender una conversación la “calidad telefónica” con un solo canal y muestras

de 8 bits es más que suficiente puesto que al muestrear a 8Khz registra frecuencias de hasta 4Khz y la información importante del habla humana está en sonidos que están dentro de ese rango de frecuencias. Eso produce un flujo de datos que es más que conocido en telecomunicaciones porque es el que se utiliza siempre para un canal telefónico: 64Kbits/s. Pero nosotros tenemos dos oídos y además captamos frecuencias de hasta (aproximadamente) 20Khz, frecuencias que muchos instrumentos musicales producen. Por tanto, para escuchar música de alta fidelidad no queda más remedio que usar “calidad CD”, con dos canales y muestras de alta resolución (16 bits). Ya hemos dicho que la calidad CD produce un flujo de datos de 1.4112Mbits/s.

Los formatos más simples de audio digital, al igual que los formatos gráficos, sencillamente guardan las muestras sucesivamente, en un formato “crudo”. Eso ocupa bastante espacio, de manera que en general los ficheros de audio guardan la información comprimida; pero algunos soportes como el CD-Audio no la comprimen (seguramente se hizo así porque en el momento de lanzar el producto era caro fabricar los reproductores con un descompresor, algo que ahora no tiene sentido, o quizás pensaron que 74 minutos de audio era más que suficiente para un solo disco). Los formatos que típicamente guardan la información digital “cruda”, y a veces comprimida con algoritmos “lossless” (Huffman, LZ, etc) son algunos de los más usados porque no producen ninguna pérdida de calidad: por ejemplo, Au, Wav, Aiff, etc. Además, existen otros formatos más complejos (basados en técnicas similares a las usadas por los formatos gráficos JPEG y el video MPEG) como por ejemplo el MP3 (que no es más que una capa de MPEG, la capa de audio) donde también se puede esteganografiar información, solo que es más complicado. Veremos un ejemplo en la parte de aplicaciones reales.

Sin embargo, esteganografiar información en los formatos más simples es prácticamente trivial: igual que en el caso de las muestras de color para las imágenes, si introducimos variaciones suficientemente pequeñas en las muestras de audio los cambios serán imperceptibles para un oído humano normal. Si las muestras son de 8 bits las amplitudes diferentes disponibles son 256 y realmente alterar el bit menos significativo no produce un cambio muy grande pero puede llegar a ser perceptible. En cambio, el audio con muestras de 16 bits tiene 65536 valores de amplitud diferentes que (definitivamente) es más de lo que un oído humano normal es capaz de distinguir. Por tanto, modificar el bit menos significativo de esas muestras no produce diferencias perceptibles. Esos son los bits que podemos aprovechar para esteganografiar la información.

Además, si tenemos en cuenta que los sonidos de altísima calidad (CD-Audio) tienen dos canales de audio a 44.1Khz y muestras de 16 bits llegamos a la conclusión de que en una señal de audio digital de estas características podemos esteganografiar (¡como máximo!) $\frac{2 \cdot 44100 \cdot 1}{8} = 11025$ bytes por segundo, solo alterando el bit menos significativo de cada muestra. Eso permite esteganografiar una cantidad de información más que respetable (decenas de megabytes) en un CD de audio. Y en ficheros de audio digital (por ejemplo Wav) de unos pocos minutos en calidad CD, podemos esteganografiar centenares de kilobytes o incluso varios megabytes.

Por supuesto todo esto son máximos teóricos, pero dado que los sistemas de audio digital tienen unos filtros pasabajas a la salida que sirven de alguna manera para suavizar las asperezas digitales, esas modificaciones introducidas por la esteganografía son difícilmente detectables por un oído humano (por muy experimentado que sea). En cualquier caso por seguridad parece recomendable no utilizar toda la capacidad del fichero y, al igual que en los ficheros gráficos, conviene elegir de forma pseudoaleatoria la secuencia de muestras que se alteran y dispersarlas al máximo por todo el fichero. Además como los datos que alteramos son los bits menos significativos y seguramente desde el punto de vista estadístico parecen ruido blanco, es conveniente que la información que esteganografiamos esté comprimida (mejor si además está encriptada) para minimizar el impacto de la esteganografía sobre el fichero, mantener sus propiedades estadísticas, etc.

2.2.4 En lugares todavía más extraños

Una vez explicado como puede esteganografiarse (mediante las técnicas más elementales) información en imágenes estáticas y en audio, es evidente que todo tipo de información digitalizada es susceptible de contener información esteganografiada porque en todos los casos existen unas muestras que pueden ser alteradas ligeramente sin grandes consecuencias. Un caso aparte son los

ficheros gráficos o de audio generados por síntesis (mediante programas como Blender, 3D Studio, Lightwave, Alias-Wavefront, etc); estos no son tan adecuados puesto que su origen algorítmico altamente determinista hace que no tengan esa componente aleatoria que tienen las informaciones digitalizadas a partir de señales del “mundo real”.

Aunque no conocemos la existencia de ninguna aplicación real que lo haga, después de ver lo que puede esteganografiarse en imágenes y en audio parece evidente que el siguiente paso es esteganografiar información utilizando video como portadora. Sobre todo, video MPEG que es el formato más común. Como veremos en la parte de aplicaciones reales, ya se han hecho programas que esteganografían información en imágenes JPEG y en audio MP3 (la capa de audio de MPEG). Además, si tenemos en cuenta que el formato MPEG es el que utilizan los DVD's y que en un DVD caben unos 17Gb de video (aproximadamente 8 horas de video de máxima calidad, con banda sonora envolvente de 5 canales en varios idiomas, subtítulos en decenas de ellos, etc) resulta tentador pensar en las posibilidades esteganográficas de ese medio. Las de un simple CD son suficientemente espectaculares.

Otro tipo de datos que parece útil para la esteganografía (pero que todavía no se ha explotado porque no es muy común) es la información topográfica. Algunos programas manejan ficheros que contienen varios gigabytes de información topográfica extremadamente precisa (pero medida del mundo real) que se utiliza para multitud de aplicaciones: confección de mapas, construcción de grandes obras de ingeniería, diseño de radioenlaces terrestres, etc. A menudo esa información se guarda en formatos con una precisión superior a la de los instrumentos de medida o incluso superior a la necesaria, y por tanto podría ser útil para la esteganografía porque sutiles cambios en ciertos valores podrían no tener consecuencias visibles.

Además de todo esto, existen aplicaciones reales que esteganografían información utilizando como portadora ficheros en formato HTML, \LaTeX , \LyX , PostScript, PDF, etc. Pero como todos estos formatos se basan en texto ASCII podemos considerarlos casos especiales de la esteganografía en textos. Sencillamente esos programas se aprovechan de algunas de las particularidades de esos formatos para hacer mejor su trabajo de ocultación, pero nada más.

2.3 En dispositivos

Una aproximación totalmente diferente es la esteganografía en dispositivos. Esta técnica no utiliza redundancias o partes aleatorias de la información que hace de portadora para introducir información secreta, sino que la “portadora” es un dispositivo. Es algo muy diferente, pero igual de válido para esteganografiar información. Simplemente se basa en la utilización de un lugar original e inesperado para esconder la información, y no utiliza codificaciones extrañas para entrelazar discretamente diferentes tipos de información, aunque es recomendable de todas formas comprimir y encriptar antes los datos secretos para que den la menor información posible si son descubiertos.

Tomemos por ejemplo un disquete. Cualquier disquete que contenga información estará probablemente organizado en unas unidades mínimas llamadas sectores, que típicamente son de 512 bytes. Cualquiera que sea el sistema de ficheros en que esté organizado el disco de forma lógica, siempre tendrá por debajo esa estructura física en sectores; además, la estructura lógica también tendrá su unidad mínima de ocupación (siempre mayor o igual que un sector) que por ejemplo en el caso de los sistemas FAT se llama cluster y en el caso de los disquetes suele ser también de 512 bytes. Pues bien, es perfectamente posible diseñar un programa que detecte cuáles son los sectores que no están ocupados en un disquete en un momento determinado, y que los utilice para esteganografiar información. Por supuesto es posible (haciendo operaciones a bajo nivel) escribir información en esos sectores sin utilizar para nada el sistema de archivos, el único peligro es que el sistema de archivos considera esos sectores como espacio libre y puede utilizarlos cuando le convenga. Pero si no se modifica el disco después de esteganografiar la información, no hay ningún problema. Además, los sectores que todavía estén libres después de la esteganografía pueden rellenarse con información pseudoaleatoria para evitar que contrasten con los que contienen información. Por supuesto al igual que en las otras técnicas esteganográficas expuestas es conveniente utilizar algún criterio pseudoaleatorio para elegir la secuencia de sectores que van a utilizarse para esconder la información, procurando la máxima dispersión por todo el espacio libre

del disco.

Existen otras técnicas todavía más sencillas pero que pueden ser útiles: hoy en día casi todo es pequeño, portátil, electrónico y digital; y suele llevar algún tipo de memoria. Los relojes, las calculadoras, las cámaras digitales, los reproductores de MP3, los teléfonos móviles y un número cada vez mayor de aparatos cotidianos de apariencia inocente tienen una creciente capacidad para almacenar información en formato digital. Eso puede ser explotado para almacenar o transmitir información sin despertar sospechas. Por ejemplo la calculadora HP-49 tiene 1.5Mbytes de memoria disponibles para el usuario: 256Kbytes de IRAM de usuario, más 256Kbytes de ERAM para almacenamiento a medio plazo de datos/programas y además 1Mb de memoria Flash, útil para almacenamiento indefinido de datos a largo plazo. Mejor todavía: tiene otro Mbyte de Flash para almacenar el Sistema Operativo y las aplicaciones, y resulta que las últimas actualizaciones disponibles para la calculadora ocupan aproximadamente un 80% del espacio reservado para ellas, dejando libre un 20% sin usar. Ese es un sitio ideal para esconder información; no es ciencia ficción: en ese 80% de espacio ocupado por las últimas versiones del Sistema Operativo y las aplicaciones ya tenemos cosas ocultas. Pulsando largas combinaciones de teclas pueden verse por pantalla decenas de fotografías del equipo de diseño de la calculadora. En lugar de ser inocentes fotografías podrían ser oscuros secretos comerciales encriptados y esteganografiados; tal vez en el fondo lo sean. Además, la mayoría de cámaras digitales y reproductores de MP3 utilizan tarjetas de memoria del tipo SmartMedia, CompactFlash o MemoryStick con capacidades interesantes (hasta 64Mbytes, de momento).

Y pensando en lugares extraños, la mayoría de las BIOS de los ordenadores y equipos electrónicos actuales están guardadas en FlashROM, de manera que son actualizables, lo cual resulta útil para añadir nuevas funciones al equipo o corregir bugs de versiones anteriores. Como la memoria es cada día más barata y la capacidad de los chips es típicamente una potencia de dos (debido al tema del direccionamiento), a veces resulta que sobra algo de memoria y para aprovecharla de alguna manera los fabricantes suelen incluir gráficos (¡o incluso sonidos!) en esas BIOS; gráficos y sonidos que se reproducen cada vez que el sistema arranca, o cuando el usuario pulsa ciertas combinaciones de teclas. En el fondo son gráficos y sonidos como otros cualesquiera, y pueden servir como portadoras esteganográficas. Y eso no es más que el principio: las mentes más retorcidas pueden intentar esteganografiar información modificando cuidadosamente el número de serie del firmware de su reproductor de CD-ROM, que seguramente también está guardado en FlashROM. Otra vez más: las posibilidades son ilimitadas.

2.4 En protocolos

La esteganografía en protocolos es algo muy interesante que todavía no se ha explotado... o si se ha hecho, ha sido en secreto. Realmente tiene tantas posibilidades que es difícil creer que se haya quedado en una buena idea. De hecho veremos una aplicación real que esteganografía información en la pila de protocolos TCP/IP; aunque lo hace de forma muy burda y a título de demostración, sirve para ver que es posible y que podrían desarrollarse aplicaciones esteganográficas muy sutiles para establecer canales de comunicación encubiertos sobre TCP/IP. Además, la idea general es extrapolable a otros muchos protocolos.

Para empezar, casi todos los protocolos de comunicaciones tienen un formato de datos muy definido. En cualquier nivel (utilizando OSI como referencia, desde el nivel físico hasta el de aplicación) nos encontramos una forma muy concreta de estructurar la información, ya sean tramas, paquetes, datagramas o segmentos, y esa estructura marca completamente la utilidad y el comportamiento del protocolo. Esas estructuras de datos tienen unos formatos muy concretos que definen exactamente el tamaño en bits de unos campos con unas funciones específicas; y normalmente es posible que haya que intercambiar diferentes tipos de mensajes, pero en cualquier caso para facilitar la implementación se aprovecha el mismo formato básico de información con unas variaciones mínimas. Eso hace que en muchas ocasiones la información viaje encapsulada en estructuras de datos (podríamos llamarlas genéricamente PDU's, por "Protocol Data Unit") que tienen varios campos con diferentes funcionalidades que no se utilizan en todas las transmisiones, pero que están ahí por homogeneidad. Esos campos son los que pueden servir para esteganografiar

información.

Por poner un ejemplo, si un determinado protocolo orientado a la conexión utiliza siempre el mismo formato de PDU para negociar el establecimiento de la conexión, para transmitir los datos y para cerrar la conexión, es probable que haya algunos campos de la PDU que indiquen el tipo de mensaje que llevan (conexión, datos o desconexión) y en función de eso otros campos tengan sentido o dejen de tenerlo. Por ejemplo (para no utilizar ningún caso concreto, son escasos y ya veremos uno en la parte de implementaciones reales) es perfectamente posible que el protocolo en cuestión tenga un campo de número de secuencia que solamente utiliza cuando está transmitiendo datos para garantizar que llegan en el orden correcto, pero que no usa estableciendo o cerrando conexiones porque entonces es suficiente utilizar otro campo con un identificador único de conexión. Entonces es posible mandar información encubierta en ese protocolo enviando sucesivas PDU's de establecimiento/cierre de conexión con los datos escondidos en el campo de secuencia, que en esas PDU's no tiene sentido pero para facilitar la implementación se mandan en cualquier caso. Por supuesto en ese caso se pierden muchas de las ventajas (quizás detección/corrección de errores, secuenciamiento, etc) que tiene enviar los datos en el campo de datos en la utilización normal del protocolo (que fué diseñado para eso y no para otra cosa), pero no aparecen dificultades insalvables.

A parte de eso, en las PDU's de diversos protocolos suelen aparecer campos etiquetados como "reservado", "experimental" o "para uso futuro". Son campos realmente pequeños, pero son campos que están ahí y se transmiten porque forman parte de las especificaciones del protocolo, y normalmente nadie se fija en ellos; ni siquiera suele estar especificado el valor que deben tomar por defecto, quedando eso a criterio del implementador. Esos campos son potenciales portadores de información esteganografiada.

2.5 Otras técnicas, y técnicas complementarias

La lista de técnicas esteganográficas es interminable. Sin embargo, algunas de ellas son muy importantes (generalmente por su simplicidad) y todavía no hemos hablado de ellas. Hay muchísimas, pero evidentemente solo podemos hablar de las que conocemos.

2.5.1 *Appending*

La mayoría de formatos de fichero (de todo tipo: ejecutables, gráficos, sonido, video, etc) toleran sin ningún problema que se añada cualquier tipo de información al final del fichero. Sencillamente los ficheros suelen estar organizados internamente siguiendo un formato estándar (por ejemplo ELF para los ejecutables, BMP para los gráficos, Wav para el sonido, MPEG para el video, etc) con sus campos perfectamente organizados, o tal vez estructurados en chunks... eso no importa ahora.

Lo realmente importante es que cuando se usan esos ficheros (en todos los casos están hechos para usarlos) el hecho de que el fichero tenga un tamaño superior al esperado y contenga datos extra al final no es relevante en la mayoría de los casos. Esto es porque si una aplicación sabe exactamente a que posiciones del fichero debe acceder para leer ciertos tipos concretos de información, puede hacerlo sin que unos pocos (o muchos) bytes extra al final afecten su correcto funcionamiento. Sencillamente, esos datos añadidos al final serán ignorados: precisamente porque "nadie" (ningún software) espera que se encuentren allí, es probable que nadie advierta su presencia y por tanto normalmente serán tolerados. Para poner un ejemplo concreto: una forma muy burda pero a veces efectiva de esteganografiar una fotografía (mejor si está muy comprimida en formato JPEG) dentro de un gran archivo de audio Wav, puede consistir sencillamente en añadir el fichero JPEG al final del fichero Wav. El audio se reproducirá correctamente y el fichero parecerá completamente normal... ¡siempre que el sentido común nos haya llevado a usar un fichero Wav mucho mayor que la fotografía, para que el tamaño de esta pase desapercibido!

Por supuesto un fichero de estas características puede resultar muy fácil de detectar (ya sea algorítmicamente o mediante simple inspección visual de un volcado hexadecimal del mismo) si se sospecha algo así, pero si el "enemigo" no se lo espera, es una técnica tan válida como la otra. Especialmente si los datos del final están encriptados, puede resultar bastante desconcertante.

Aunque parezca algo trivial, existe una aplicación que implementa esta técnica y que por su extrema simplicidad no veremos en la parte de implementaciones reales: se trata del *PGE (Pretty Good Envelope)*.

2.5.2 Introducción de “basura”

Cuando vemos por dentro los formatos de fichero diseñados por Microsoft (de todo tipo, pero especialmente DLL y DOC) no podemos evitar pensar que se trata de formatos esteganográficos: tan poca información perdida en unos ficheros tan grandes... tiene que ser para despistar al “enemigo”. Sin duda la técnica consiste en desconcertar al atacante, que tiene grandes dificultades para localizar la información que busca entre tanta información inservible.

Bromas aparte, esa técnica puede resultar muy útil y puede complementar a muchas técnicas esteganográficas ya vistas: la información esteganografiada puede ocultarse mucho mejor si está entre mucha información similar e inútil. Esto puede hacerse a varios niveles. Por ejemplo, si antes de proceder a la ocultación de una información secreta (ya comprimida y encriptada) la mezclamos cuidadosamente con “basura” pseudoaleatoria, dificultamos mucho su detección. De la misma manera, puede resultar útil añadir algo de “basura” pseudoaleatoria antes de proceder a la encriptación, para dificultar al criptoanálisis. Y sin duda, las “portadoras” que llevan información esteganografiada estarán mucho más protegidas si las guardamos o las transmitimos junto a otras portadoras similares sin ninguna información oculta.

2.5.3 Eliminación de redundancias y patrones fijos

Después de ver el añadido de “basura” pseudoaleatoria a la información secreta, tenemos la técnica “simétrica”, por decirlo de alguna manera. Digamos que es la que elimina todo tipo de información superflua antes de la esteganografía. Por ejemplo, si la información secreta que queremos esteganografiar está en un fichero comprimido con *RAR* (un excelente compresor de EUGENE ROSHAL) deberíamos tener en cuenta que ese formato de fichero siempre empieza con la cadena “Rar!” y tal vez sería conveniente eliminar esos cuatro caracteres antes de continuar. El receptor podría reponerlos, si sabe en qué formato está el fichero.

Si utilizamos continuamente los mismos formatos, hay que tener en cuenta estas cosas. De hecho existe un programa (también muy simple y que no mencionaremos más) que se llama *Stealth*; no es más que un filtro para *PGP (Pretty Good Privacy)* que lo que hace es eliminar de la cabecera todas las partes superfluas (las que puede reponer el receptor) para dejar los ficheros encriptados/firmados con PGP en un formato más apto para la esteganografía.

3 Estándares esteganográficos

3.1 No hay ninguno

Aunque hemos encontrado algunas patentes sobre técnicas esteganográficas, no hemos podido encontrar ningún estándar sobre esteganografía. Ninguna técnica, ningún algoritmo esteganográfico merece el calificativo de “estándar”. Ni ITU-T, ni ANSI, ni ISO, ni DIN, ni UNE... ni nada. Ninguna organización de estandarización más o menos conocida parece saber nada de la esteganografía.

A veces se trata el tema en revistas científicas o técnicas, se han publicado artículos para el IEEE y se han hecho algunos workshops (especialmente sobre la parte más comercial de la esteganografía: el *watermarking*) pero nada más.

3.2 Por qué?

Básicamente por el motivo que hemos mencionado en la introducción: ya que gran parte del éxito de la esteganografía depende del secreto, no tiene demasiado sentido adoptar un estándar esteganográfico. Veamos con más detalle los motivos.

Lo mejor que puede pasar es que el “enemigo” ignore por completo que usamos esteganografía. En ese caso, las portadoras de apariencia inocente no despertaran sospechas y llegaran a su destino sin problemas. Si no estamos en el mejor caso y el “enemigo” sospecha que utilizamos esteganografía, ayuda bastante que ignore exactamente que técnica utilizamos. Si además estamos en un caso malo y el “enemigo” no solo sabe que utilizamos esteganografía sino que conoce exactamente la técnica utilizada, lo único que puede protegernos ya es la encriptación del mensaje secreto y en ese caso encontramos que la esteganografía ha resultado inútil y superflua.

Incluso usando las técnicas esteganográficas más seguras en que conocer la técnica utilizada no es suficiente sino que hace falta conocer algún tipo de parámetro (típicamente la semilla de un generador de números pseudoaleatorios, o alguna cosa similar) tenemos problemas si el enemigo intercepta una portadora y sabe con absoluta certeza que contiene información esteganografiada, porque precisamente hemos utilizado esteganografía para mantener en secreto la existencia de esa información. En cierta manera en esos casos el ataque contra el algoritmo esteganográfico para extraer la información (cifrada) que contiene la portadora puede considerarse como un paso previo al ataque de criptoanálisis. Parte del secreto (la propia existencia de un secreto) ha sido ya descubierto.

Por estos motivos, si en un estándar se explica una técnica esteganográfica muy bien diseñada con todo lujo de detalles y luego alguien anuncia que utiliza ese estándar, está anunciando públicamente dos cosas: que tiene información secreta cuya existencia desea ocultar, y que intenta hacerlo siguiendo la técnica especificada en el estándar. Y habrá que asumir que el “enemigo” seguramente sabrá eso, con lo cual nos encontramos en una situación que no tiene ningún sentido: hacer eso equivale a prescindir de la esteganografía y utilizar directamente la criptografía.

3.3 Patentes

Al parecer algunas personas han tenido la idea de patentar técnicas bastante sofisticadas relacionadas con la esteganografía. Por ejemplo (no sin dificultades) hemos encontrado la *United States Patent Number 5,613,004 "Steganographic Method and Device"*.

En este documento se analizan y se definen de forma clara y breve la criptografía y la esteganografía en general, a modo de introducción. Después se analizan los problemas y los inconvenientes de utilizar solamente criptografía para proteger copyrights/propiedades intelectuales, poniendo como ejemplo algunos sistemas de televisión por cable/satélite.

Finalmente se registra un “invento” consistente en un dispositivo llamado “Stega-Cypher” que combinando criptografía y esteganografía pretende solucionar todos estos problemas. Básicamente acaba describiendo un mecanismo de *watermarking* con ligeras modificaciones. Si ha llegado a utilizarse alguna vez, no lo sabemos; intuimos que no, pero la patente está ahí.

4 Aplicaciones de la esteganografía

Como ya hemos dicho, la esteganografía es un arte (y una ciencia). Por eso la creatividad, la imaginación y la originalidad son muy importantes para que funcione. Decimos eso porque puede usarse para muchos fines distintos, puede hacerse prácticamente en cualquier sitio, de cualquier forma, utilizando multitud de técnicas diferentes... En este apartado lo que pretendemos es mostrar como la esteganografía (o variantes) puede llegar a ser algo útil en el mundo real. Son solo unos pocos ejemplos, pero escogidos para que sean suficientemente significativos.

4.1 Enviar mensajes cifrados donde estan prohibidos

Ese es uno de los ejemplos más claros que podemos encontrar. En Francia para usar criptografía hace falta obtener la autorización del Gobierno. Es decir, que “por defecto” los particulares no pueden utilizar la criptografía según sus necesidades (hay algunas excepciones, pero muy poco significativas). O, dicho de otra manera, se considera que aquel que cifra sus documentos tiene algo que ocultar. Utilizando la analogía de PHIL ZIMMERMANN (*el autor de PGP*) se prohíbe

la utilización de sobres y se impone el uso de postales en su lugar, para la correspondencia de todo tipo. En una situación como esa, es evidente que la esteganografía es la solución: si puede considerarse que los mensajes cifrados son sistemáticamente interceptados y su mera existencia constituye un delito, lo mejor es esteganografiarlos en alguna información inocente (fotografías, sonidos, etc) para no llamar la atención.

En China la situación es similar aunque algo más peligrosa: sabiendo que allí el gobierno controla el acceso a Internet y que han ejecutado a unos hackers solo por robar unos miles de dólares en un banco virtual, es fácil imaginarse lo que puede pasarle a alguien a quien la policía encuentre en posesión de ficheros encriptados. Si ese “alguien” no quiere (o no puede) descodificar esa información, tendrá problemas muy serios. Y en cualquier caso probablemente los tendrá. También aquí la esteganografía puede ser la solución. Pero es conveniente que nos fijemos también en el apéndice (sección 6) de este documento porque en esos casos también puede ser necesario esconder las propias herramientas esteganográficas y de cifrado, ya que la mera posesión de esas herramientas puede ser un auténtico problema.

Finalmente, tenemos el caso de *USA*. Es un país democrático y libre... pero prohíbe la exportación (fuera de *USA* o *Canadá*) de criptografía “fuerte” en formato electrónico y sus leyes comparan eso con las municiones. No prohíben explícitamente el uso interno de la criptografía, de la esteganografía o de cualquier otra técnica, pero ha habido intentos (el famoso chip *Clipper*) desde el Gobierno (o mejor dicho, ciertas agencias del Gobierno en especial la NSA y similares) para controlar y monitorizar las comunicaciones cifradas de todo el mundo. Por otra parte tenemos cosas como el proyecto *Echelon*. Afortunadamente los legisladores americanos todavía no se han atrevido a ejercer la censura prohibiendo la exportación de libros, de manera que esa criptografía “fuerte” que no puede exportarse en formato electrónico, sí que puede exportarse... en papel. Por supuesto eso es un trabajo inmenso porque si bien imprimir el código fuente de los programas en numerosos y pesados volúmenes no es nada complicado, sí lo es realizar el proceso inverso fuera de USA y escanear todos esos volúmenes hasta reconstruir el código fuente original sin errores. Ese proceso parece que es la única posibilidad legal de sacar de USA esa tecnología, pero teniendo en cuenta que fuera de USA existe tecnología igual o incluso superior, y que existe una forma alternativa (laboriosa pero totalmente legal) de realizar la exportación, resulta que la conclusión lógica es que esas leyes son una estupidez sin sentido. En ese caso, la esteganografía parece un medio más práctico, rápido, lógico y quizás más seguro de sacar ciertos programas de USA.

4.2 Digital Watermarking: prácticamente, la única aplicación comercial

Una aplicación de la esteganografía que está teniendo cierto éxito y que es tenida en cuenta por importantes marcas comerciales de software es el “Digital Watermarking” (la traducción libre sería “marcas de agua digitales”). Básicamente la idea es proteger ciertos documentos con copyright (fotografías, dibujos, música, etc) de la copia ilegal. Es bien sabido que típicamente la información digital puede copiarse sin ninguna pérdida de calidad, y eso incluye (de forma indiscriminada) todo tipo de información incluyendo la que tiene derechos de autor. La idea original y básica del *watermarking* es utilizar la esteganografía para esconder algún tipo de información de copyright o de información sobre el autor o propietario en las imágenes o en el audio a proteger, de forma que si alguien los reproduce sin permiso reproducirá implícitamente esas “marcas” y por tanto prácticamente estará confesando su culpabilidad.

Esta técnica va un poco más allá y de hecho asimila las obras en bits a las obras en átomos, utilizando la terminología de NICHOLAS NEGROPONTE. Vamos a aclarar ese punto: resulta que las obras en átomos, las obras materiales, pueden reproducirse; pero el proceso suele implicar una pérdida de calidad o como mínimo una modificación que las hace diferentes en algo. El problema de las obras digitales es que las copias son idénticamente iguales e indistinguibles de las originales. Eso puede arreglarse con el *watermarking*, ya que esas marcas pueden finalmente eliminarse de los documentos que protegen... a costa de una ligera pérdida de calidad o de unas ligeras modificaciones que hacen diferente la copia del original. Veremos eso en el apartado de aplicaciones reales, ya que existen programas para probar la robustez de los distintos tipos de *watermarking* y que intentan eliminarlos mediante técnicas realmente inteligentes.

A parte de la protección de derechos de autor, propiedad intelectual, y demás, la esteganografía (el *watermarking* en este caso) puede usarse para autenticar ciertos tipos de documentos. En general, los documentos oficiales (por ejemplo los impresos por la *FNMT, Fábrica Nacional de Moneda y Timbre*) suelen tener unas características especiales que dificultan su falsificación. Ciertos documentos digitales análogos a los papeles de toda la vida podrían ser gráficos, y para autenticarlos podría usarse el *watermarking*. Eso sería especialmente útil en documentos que deben poder ser copiados sin problemas pero que solo pueden ser emitidos por alguna autoridad. Por ejemplo, algún tipo de certificado o título oficial podría darse en forma de imagen para su uso digital (digamos, para su publicación en una página Web), y para mostrar su autenticidad esa imagen podría llevar esteganografiada la firma digital de la FNMT sobre algunos de los datos del titular. Sobre el uso del *watermarking* en esquemas de autenticación (en concreto para tarjetas de crédito), lo mejor que hemos encontrado está en la página Web de EIJI KAWAGUCHI, coautor de la “*BPCS-steganographic technique*” (ver bibliografía).

4.3 Simple pasatiempos (?)

Sin duda una de las principales aplicaciones de la esteganografía es el de simple pasatiempos: se trata de algo que puede llegar a ser muy divertido. Además, cualquier profesional que trabaje habitualmente con sistemas de almacenamiento, procesado o transmisión de la información debería conocer las posibilidades que ofrece la esteganografía, aunque solo fuese por haber jugado alguna vez con técnicas esteganográficas.

Sobre todo decimos esto porque la esteganografía es algo que afecta muy directamente a la seguridad de los sistemas de todo tipo, por su enorme capacidad de encubrir la información. Por poner un ejemplo, utilizando algunas de las técnicas de esteganografía (y “*bouncing*”, o “rebote”) en TCP/IP que veremos más adelante podría construirse un nuevo tipo de troyano tremendamente difícil de detectar que además podría sacar sin problemas información secreta de un sistema escondiéndola en paquetes IP de apariencia realmente inocente que atravesarían la mayoría de firewalls existentes sin mayores problemas. Solo por eso vale la pena que los administradores de sistemas se dediquen un poco a jugar con la esteganografía y a ver sus posibilidades.

4.4 Nuestras sugerencias, y ejemplos ficticios

4.4.1 Rizar el rizo

Antes que nada queremos mencionar un método que creemos que se aplica bastante poco (si es que se ha aplicado alguna vez) y que además parece que no consta en ningún documento sobre seguridad de entre todos los que han pasado por nuestras manos. En un increíble acceso de paranoia y de retorcimiento intelectual, se nos ha ocurrido que podría ser útil y *muy* seguro esteganografiar la información en alguna portadora relativamente “sospechosa” para luego encriptarla y transmitirla habitualmente como si fuese un mensaje encriptado normal.

Consecuencias de la utilización de este método: si un supuesto atacante consigue algún día (después de invertir esfuerzos, tiempo y dinero en ello) romper el sistema criptográfico y por tanto finalmente consigue acceso libre a la información que estaba encriptada... ¿realmente intentará analizarla para ver si contiene información esteganografiada, para ver si hay algo más escondido ahí?

4.4.2 Mensajes secretos a través de web, ftp, e-mail o news.

Es posible esconder información en una imagen o en audio digital. Además, ese tipo de información multimedia es cada vez más frecuente en Internet y la verdad es que ya no llama la atención... aunque el e-mail de texto sigue siendo uno de los servicios más utilizados. Las consecuencias son inmediatas y aplastantes: podemos utilizar una información que circula libremente y sin despertar sospechas para esconder información tan sensible que queremos poder ocultar su existencia. Encontramos información multimedia en las páginas Web, en los servidores de FTP, en los *attachments* de muchos mensajes de correo electrónico y por supuesto en las *news*.

Es decir: suponiendo que alguien esté escuchando nuestras comunicaciones o incluso -quién sabe- espiando remotamente nuestro disco duro mediante un sofisticado ataque *TEMPEST* o sencillamente con un troyano, los gráficos y los sonidos aparentemente inocentes no llamarán mucho su atención porque es normal manejar información de ese tipo y normalmente es lo que parece. Gracias a la esteganografía, no siempre es así.

Pongamos un ejemplo; creíamos que esta idea era nuestra pero por lo visto EIJI KAWAGUCHI ya tiene publicado algo muy similar en su Web. La idea es la siguiente: podríamos crear una página Web que diariamente (o semanalmente) regalara a los visitantes interesados una imagen grande y libre de derechos de autor, útil para poner de fondo en el escritorio. Incluso podríamos poner algún banner publicitario para dar credibilidad a la página y justificar de alguna manera su existencia mediante un pequeño beneficio económico. La página tendría más o menos éxito, y la imagen de fondo (por ejemplo un gran BMP de 1024x768 pixels, a 24 bits de color, que podría ocupar fácilmente 1 MB utilizando compresión) sería de dominio público. Sin embargo, hay que recordar que una imagen de esas dimensiones puede almacenar cantidades importantes de información esteganografiada (desde unos 100KB con técnicas clásicas hasta unos 500KB con el avanzado *BPCS* de KAWAGUCHI, valores que aproximadamente se duplicarían comprimiendo antes la información secreta -si esta fuese textual-). Y nada nos impediría utilizar esa imagen para esteganografiar información de cualquier tipo que el receptor podría recuperar tranquilamente después de bajarse la imagen como cualquier otro visitante de la Web.

Pues bien, variantes de esta técnica utilizando otros medios (news, e-mail, etc) u otras portadoras (audio, poesía, lo que sea) son totalmente factibles y además muy seguras, sobre todo si (como ya hemos dicho) la información es comprimida y encriptada antes de la esteganografía.

Además, hay que tener en cuenta que en ningún momento hace falta dedicar una portadora a un solo mensaje secreto: utilizando técnicas más o menos avanzadas es perfectamente posible esconder varios mensajes de forma independiente en una misma portadora, como veremos al analizar el excelente *OutGuess* de NIELS PROVOS; este programa utiliza sofisticados códigos correctores de errores (concretamente códigos de GOLAY) para tolerar colisiones entre diferentes informaciones esteganografiadas en una misma imagen.

4.4.3 Johnny Mnemonic

Normalmente no habríamos puesto este ejemplo en un trabajo universitario, pero dado que unos profesores consideraron adecuado utilizarlo en un examen de *Fundamentos de Computadores* hace algunos años, hemos creído que nosotros también podíamos sacarle provecho.

La idea es sencilla, y es un ejemplo más de posibles aplicaciones de la esteganografía (aunque en el presente no pueda llevarse a la práctica y es cietamente dudoso que algo semejante pueda hacerse en el futuro). Básicamente en esa película el protagonista es capaz de cargar varios GigaBytes de datos en su cerebro gracias a un implante quirurgico muy avanzado; de hecho hace de correo y transporta importantes secretos comerciales robados a una importante compañía; los datos forman parte de él y se moverían con él por entre el resto de personajes de la película sin problemas, si no fuese porque debido a las circunstancias hay gente que sabe que lleva esa información y desea cortarle la cabeza para obtenerla. El resto del argumento no nos interesa ahora, porque ya hemos visto la idea que queríamos destacar de la película: el protagonista lleva información oculta en su cabeza. ¡Eso es esteganografía!

5 Implementaciones reales (software)

Veremos aquí una pequeña muestra de las aplicaciones esteganográficas disponibles, cuya documentación constituye una de las mayores fuentes de información que puede encontrarse actualmente sobre esteganografía. Después de ver este apartado, resulta evidente que un “enemigo” que pretenda interceptar todas las comunicaciones existentes encontrará en la variedad de aplicaciones esteganográficas un verdadero problema.

5.1 Software para Unix

5.1.1 Jsteg (patch para jpeg-v4)

Los ficheros gráficos en formato JPEG parecen muy poco indicados para la esteganografía, porque su objetivo es siempre obtener unos ratios de compresión tan grandes como sea posible, una vez fijada la calidad deseada. Para ello recurren a técnicas de compresión “lossy” (pierden información) y la verdad es que debido a eso introducen ruido y distorsión en la imagen, cosa que en principio destruiría la información esteganografiada si la escondiésemos antes de la compresión como se hace normalmente en los formatos gráficos que usan compresión *lossless*.

Sin embargo, este software aprovecha que el formato JPEG no utiliza en todo momento compresión *lossy*. En realidad la compresión tiene lugar en dos pasos: primero, realiza sobre la imagen una DCT (*Discrete Cosine Transform*) y un proceso de cuantización sobre los coeficientes frecuenciales resultantes que elimina tanta información como sea posible sin bajar de los umbrales de calidad exigidos por el usuario; segundo, comprime el resultado mediante un algoritmo *lossless*, el de Huffman. Entre esos dos pasos es posible introducir información secreta que podrá ser recuperada sin problemas, precisamente porque Huffman no pierde información. Concretamente es posible esconder bits de información secreta en los bits más bajos de los coeficientes frecuenciales antes de pasarlos al algoritmo de Huffman, sin que los efectos visuales sean apreciables. Si la información esteganografiada parece aleatoria, resultará extremadamente difícil de detectar.

Esto en realidad es muy práctico porque el formato JPEG es muy común y no despierta sospechas en ninguna parte. Además, si bien es cierto que el proceso de compresión *lossy* introduce ruido y distorsión en la imagen que destruiría cualquier información oculta antes de ese proceso, también es cierto que el proceso inverso (la regeneración de la imagen a partir de los coeficientes frecuenciales que han pasado por la cuantización) hace lo mismo. Eso hace muy difícil distinguir en la imagen entre el ruido producido por información esteganografiada y el ruido producido por una cuantización de baja calidad.

5.1.2 Stext 1.0

Este pequeño programa esteganografía información en textos ASCII mediante una técnica de generación de portadora. Es decir, que a partir del mensaje secreto, unos diccionarios de frases y unos conjuntos de reglas simples es capaz de generar un texto (en inglés) aparentemente normal. Por lo menos desde el punto de vista sintáctico, ortográfico, gramatical e incluso estadístico, el texto parece normal. Evidentemente el programa no es inteligente y desde el punto de vista semántico el texto deja mucho que desear, pero para notar eso normalmente hace falta un observador humano.

La ventaja es que el texto ASCII simple es una información muy corriente en todas partes y llama menos la atención que cualquier otra cosa. Además, es muy fácil que un atacante decida que es suficiente analizarlo de forma automática mediante software, que normalmente pasará por alto la información oculta.

El funcionamiento es simple. En general, el programa genera nuevos textos (que contienen información secreta) a partir de textos “inocentes” ya existentes que le proporciona el usuario. Básicamente, codifica el mensaje secreto en la longitud de las palabras. Por ejemplo, en el modo de codificación de un bit por palabra se considerará cero si la palabra tiene longitud par y uno si tiene longitud impar; de la misma forma es posible codificar 2, 4, ... bits por palabra cogiendo la longitud de la palabra módulo 4, 16, ... Pero para ello hacen falta unos diccionarios realmente completos. No hay que olvidar que el programa trabaja con frases enteras y su trabajo es encontrar en los diccionarios las frases adecuadas que se ajusten a la secuencia de bits de la información secreta, para luego construir el resultado final. A parte de todo esto, el programa tiene unas reglas bien definidas para delimitar las frases y para tratar sin ambigüedades cosas como los tabuladores, los finales de línea, etc.

Este programa no incluye encriptación de la información secreta antes de la esteganografía, pero cuenta con un sencillo sistema de “scrambling” que utiliza como clave una máscara de bits que se superpone repetidamente al texto de forma que solo se tienen en cuenta los caracteres que tienen superpuesto un uno. Esto no protege la información pero dificulta el “estegoanálisis” :-)

5.1.3 StegParty 0.2

Este programa también esteganografía información en textos, pero no genera la portadora sino que esconde el mensaje secreto alterando sutilmente una portadora ya existente, según unas reglas de equivalencias que además son configurables por el usuario. El resultado es mucho más legible y en general tiene una apariencia mucho más inocente: el método es, sin duda, más seguro. A cambio, necesita grandes cantidades de texto para esconder mensajes relativamente pequeños: según el autor del programa, con las reglas básicas que incluye es posible esteganografiar unos 4KB de información en la versión electrónica (*Proyecto Gutenberg*) de la obra “*Through The Looking Glass*” de LEWIS CARROLL.

Otra vez más, el funcionamiento es (a pesar de su originalidad) bastante simple. Resumiendo, lo que tenemos en el diccionario (también llamado “fichero de reglas”) son expresiones comunes y varias formas equivalentes de escribirlas. El programa sencillamente las busca en el texto y realiza sustituciones por expresiones equivalentes (o no) en función de los bits del mensaje secreto.

Vale la pena explicar la forma de leer los bits del mensaje secreto porque es común a muchos otros programas de esteganografía: en general considera que el mensaje secreto ' x ' es un gran número binario de ' m ' bits. Cuando tiene la oportunidad de esconder ' n ' bits, esconde el valor correspondiente a ' $x \bmod n$ ' y seguidamente hace la asignación ' $x = \frac{x}{n}$ '. En este caso el número ' n ' de bits que esconde en cada ocasión se basa en el número de sinónimos que tiene la expresión que está tratando en este momento, y en su ordenación en el fichero de reglas. Visto de esta manera, el fichero de reglas actúa como “clave” del mecanismo esteganográfico. Sin embargo el programa no proporciona criptografía sólida para el mensaje secreto.

Como hemos dicho, esta técnica es muy segura pero poco eficiente; en cualquier caso la calidad del fichero de reglas es muy importante para la seguridad y la eficiencia del método.

5.1.4 StegHide 0.3

Este programa no supone ninguna novedad, como la mayoría de programas disponibles... pero alguno había que poner de este tipo porque son los más comunes. Este programa esteganografía información mediante las técnicas clásicas alterando los bits bajos de las muestras en ficheros que contienen algún tipo de información digitalizada “cruda”. Concretamente, este programa soporta el formato gráfico BMP y los formatos de audio Wav y Au, todos ellos con diferentes tipos de compresión *lossless*.

Este programa (y otros muchos de similares características) ofrece un tipo de esteganografía totalmente válido pero bastante sencillo técnicamente, y en general poco innovador. Por este motivo, este tipo de programas suelen incluir algún tipo de “extra” de seguridad más o menos original y que los hace minimamente atractivos para los usuarios. En el caso que nos ocupa, el programa es capaz de encriptar la información secreta antes de esteganografiarla, y lo hace con un algoritmo considerado bastante seguro: *Blowfish*, desarrollado por BRUCE SCHNEIER. Además hace una gestión inteligente y segura de la clave ya que no la utiliza directamente para encriptar sino que la procesa antes con el algoritmo de *hash MD5*. Para finalizar, distingue la cabecera (que también esteganografía y que guarda entre otras cosas la longitud del mensaje) de los datos secretos propiamente dichos y permite encriptarlos por separado con claves diferentes (no hablamos hasta ahora de cabeceras y longitudes, pero es evidente que si esteganografiamos información en un fichero deberemos saber de alguna manera la longitud de los datos que vamos a extraer: puede introducirla el usuario, pero lo más práctico es esconderla junto con los datos, al principio, en un campo de dimensiones bien definidas).

A parte de las particularidades de este programa, los demás de su clase (esteganografía muy básica) llevan extras más o menos espectaculares (sobre todo los que tienen un *front-end* gráfico) tales como borrado “seguro” de ficheros, varios algoritmos criptográficos y de hash para elegir, protocolos de intercambio de claves bastante avanzados, o incluso editores de texto que usan tipos de letra y modos gráficos diseñados para minimizar la emisión de radiaciones del monitor que posibilitarían un ataque remoto *TEMPEST*. Aunque respecto a este tipo de ataques, lo mejor seguramente es ferrar de plomo la habitación... o como mínimo el ordenador, incluyendo los cables

que tienen la desagradable costumbre de actuar de antena accidentalmente.

5.1.5 OutGuess 0.13b

A diferencia del anterior, este programa es realmente innovador e inteligente (desde el punto de vista de las técnicas esteganográficas, porque desde un punto de vista más general el programa anterior está bastante bien). Es un programa esteganográfico genérico que está preparado para utilizar como portadora cualquier formato de fichero, con cualquier tipo de información; solo hace falta proporcionarle lo que el autor llama “*handler*” que no es más que una especificación del formato en cuestión. El programa originalmente viene con *handlers* para los formatos gráficos PNM y JPEG. Y la verdad es que los *handlers* no son la auténtica novedad de este programa porque en realidad son difíciles de utilizar y poco prácticos.

Sin embargo, el autor de este programa ha tenido en cuenta varios aspectos importantes pero algo enrevesados de la esteganografía que le han llevado a soluciones muy buenas y a aproximaciones bastante innovadoras. Algunos de los interesantes razonamientos que aplica al diseño de su programa son:

- Hay que tener en cuenta que la información esteganografiada en último término es detectable: el receptor va a extraerla, de manera que tal cosa es posible y un “enemigo” con suerte y mucha habilidad e inteligencia podría llegar a hacerlo. En cualquier caso las portadoras pueden llegar a delatar la información secreta: por ejemplo un análisis frecuencial sobre una imagen portadora puede llegar a revelar patrones extraños e impropios de una imagen digitalizada normal que indiquen la presencia de información esteganografiada. En general la información esteganografiada es más fácilmente detectable cuantos más cambios se introduzcan en la portadora. La sorprendente solución de NIELS PROVOS a este problema consiste en elegir de entre una gran variedad de portadoras disponibles la que mejor se ajusta al mensaje secreto que queremos enviar, teniendo como criterio de elección que el número de cambios que haya que realizar sobre esa portadora sean mínimos. Además para cada portadora elige la secuencia de cambios de forma pseudoaleatoria, e intenta encontrar el valor óptimo de la semilla del algoritmo para que el número de cambios también sea mínimo. Todo eso es costoso en tiempo y en potencia de cálculo, pero con su programa demuestra que es factible.
- Existe una propiedad muy deseable que es la “*plausible deniability*”. Ésta puede conseguirse esteganografiando de forma independiente un número arbitrario de mensajes secretos diferentes en la misma portadora de manera que extraer uno (o varios) de ellos no revela nada sobre los demás y es posible negar (de forma convincente) la existencia de ciertos mensajes secretos. Así en principio un atacante no puede estar seguro de si ha obtenido todos los mensajes secretos aunque haya conseguido extraer algunos de ellos. La solución de NIELS PROVOS consiste en utilizar un complejo algoritmo para bloquear y proteger de futuras modificaciones los bits que ya han sido usados para esteganografiar otros mensajes, y además usa un código corrector de errores de GOLAY (23, 12, 7) para tolerar algunas colisiones en los bits bloqueados. Y no es precisamente una mala solución: los códigos de GOLAY (23, 12) son los únicos códigos binarios perfectos multicorrectores de errores que se sabe que son capaces de corregir cualquier combinación de 3 ó menos errores aleatorios en bloques de 23 bits.

A parte de esas inteligentes observaciones, el diseño de este programa permite encriptar los mensajes secretos antes de esteganografiarlos. Aunque no especifica con qué algoritmo criptográfico lo hace, en la documentación dice que el programa utiliza el generador de números aleatorios *Arc4* del *Unix OpenBSD* y además el algoritmo de *hash MD5*. Es posible que entre otras cosas utilice alguno de estos algoritmos para generar un stream pseudoaleatorio con la intención de cifrar la información, y en ese caso su cifrado no es el mejor precisamente, pero para dar una apariencia aleatoria al mensaje secreto parece suficiente.

5.1.6 StirMark 1.0

Este programa es el más extraño de entre todos los que hemos encontrado. Ni siquiera se trata de software esteganográfico, ni de *watermarking*. La verdad es que es un programa diseñado básicamente para romper los esquemas de *watermarking* dañando al mínimo las imágenes. O, dicho de forma algo más diplomática (como hace el autor): es un programa diseñado para comprobar la fortaleza de los sistemas esteganográficos y de *watermarking* en general.

Este programa básicamente intenta introducir pequeñas modificaciones en las fotografías digitales con la intención de que cualquier información esteganografiada que puedan contener se pierda, todo ello degradando mínimamente la calidad de la imagen. En cierta manera no es un problema que la información secreta esteganografiada se pueda perder ante este tipo de modificaciones, porque eso simplemente significa que si la portadora cae en manos equivocadas es más probable que no llegue a conocerse nunca su verdadera naturaleza. Pero en los algoritmos de *watermarking* eso es algo casi imperdonable: la información en estos casos debería ser esteganografiada con técnicas muy robustas y resistentes a modificaciones, de manera que para eliminarla hubiera que introducir grandes alteraciones que provocaran una degradación importante de la calidad de la imagen. Para nuestra sorpresa, el autor del *StirMark* afirma que con su programa ha conseguido eliminar las “marcas de agua” hechas con **todos** los paquetes comerciales de *watermarking* que ha podido encontrar, con mínimas pérdidas de calidad. Sin duda eso indica que habrá que investigar para encontrar técnicas más robustas frente a este tipo de ataque para proteger el copyright de las imágenes.

Veamos como funciona este maravilloso programa: básicamente simula por software un proceso de “*resampling*”. Es decir, introduce los mismos cambios que experimentaría una imagen si la imprimieramos con una impresora de altísima calidad y luego la capturásemos de nuevo con un escáner de altísima calidad. En primer lugar aplica una ligera distorsión geométrica a la imagen (la escala, la desplaza, la rota y la deforma de manera aleatoria pero dentro de unos márgenes casi inapreciables a simple vista, utilizando algoritmos similares a los que utilizan los programas de retoque fotográfico). Eso suele ser suficiente para destruir cualquier “marca de agua”, pero el *StirMark* sigue su ataque: después se simula un proceso de muestreo mediante un algoritmo de interpolación, y finalmente se aplica una función de transferencia que introduce una pequeñísima cantidad de ruido distribuido entre todas las muestras (para simular las imperfecciones de los pequeños conversores digital/analógico y analógico/digital que normalmente encontramos en los dispositivos electrónicos que trabajan con imágenes).

La verdad es que con los parámetros que toma por defecto la imagen resultante tiene una calidad tan alta que resulta difícil imaginarse un hardware (impresora y escáner) suficientemente bueno para conseguirla en un proceso real de *resampling*, pero la aplicación reiterada del programa o la utilización de unos parámetros que provoquen alteraciones cuantitativamente mayores a las que se introducen por defecto si que puede llegar a dar como resultado una imagen que parezca que ha sido impresa y luego escaneada. Sin embargo normalmente una simple pasada del algoritmo con los parámetros que producen pérdidas mínimas de calidad (prácticamente inapreciables para el ojo humano) suele ser suficiente para eliminar cualquier “marca de agua” hecha con el software actualmente disponible. La única limitación del *StirMark* es que solo trabaja con bitmaps, con ficheros que contienen información gráfica sin procesar (ya sea comprimida con algoritmos *lossless* o directamente “cruda”) y no con formatos gráficos más avanzados (y extendidos) como puede ser el JPEG.

5.1.7 GifShuffle, Snow y el cifrador ICE

Hemos incluido en la misma sección estos dos programas (*GifShuffle* y *Snow*) porque utilizan ambos el mismo algoritmo de encriptación (*ICE*) que además ha sido diseñado por su propio autor. Trabaja con bloques de datos de 64 bits y la clave puede ser de longitud variable (hasta 1024 bytes); no deja de ser curioso que el autor lo use siempre en CFB de 1 bit con un extraño IV inicializado a la clave cifrada por ella misma. . . Normalmente conviene desconfiar de algoritmos criptográficos más o menos desconocidos que han sido diseñados por gente más o menos desconocida, porque eso

básicamente significa que han sido muy poco probados, que han sufrido muy pocos ataques y que los “grandes” criptoanalistas tal vez ni siquiera los han visto. Aparte de eso, ambos programas utilizan un ineficaz algoritmo de compresión (afortunadamente desactivable) basado en Huffman que usa unas feas tablas precalculadas y optimizadas para comprimir texto en inglés. Su propio autor reconoce que es “rudimentario”. Pero todo esto no nos preocupa ahora porque estamos hablando de esteganografía y lo que queremos es ver como funcionan estos dos programas desde ese punto de vista:

1. Empecemos por *GifShuffle*. Este programa esteganografía información dentro de ficheros gráficos en formato GIF (puede que signifique “Graphics Interchange Format”). Como es sabido, estos ficheros no son aptos para fotografía digital porque soportan como máximo bitmaps de 256 colores y además tienen problemas de copyright porque usan compresión LZW; por todo ello, este formato caería en el olvido si no fuese porque tiene una variante “animada” que se usa extensamente sobre todo en el diseño de páginas Web. De hecho tiene más fama que el formato que se diseñó con la intención de sustituirlo, un formato muy superior en prestaciones llamado PNG (algo así como “Portable Network Graphics”).

Como ya hemos dicho en la parte de técnicas básicas, los bitmaps con pocos colores utilizan una paleta. Eso significa que al principio tienen información ordenada (normalmente RGB de 24 bits, 3 bytes por cada entrada de color) sobre los colores que van a utilizar y luego viene la información de cada pixel, que no son más que referencias a los índices de los colores de la paleta. Por supuesto existen excepciones como las imágenes monocromas o las de escalas de grises, que tienen sus formatos concretos bien definidos y no suelen necesitar paleta. *GifShuffle* trabaja con gráficos GIF de 256 colores que tienen por tanto una paleta de $3 \cdot 256 = 768$ bytes y unos datos con un byte por cada pixel, que se refiere a uno de esos colores.

La técnica esteganográfica es realmente original y puede resultar útil en ocasiones, ya que *no produce (en absoluto)* ninguna modificación visible en la imagen. Básicamente lo que hace es codificar la información secreta en la paleta, reordenando los colores que la forman y luego reajustando todas las referencias a los distintos colores para que la imagen tenga un aspecto *identicamente igual* al original. Por supuesto esta técnica permite esteganografiar poca información, pero es muy segura y a veces puede ser suficiente. Si consideramos que partimos de una paleta de 256 elementos (¡diferentes!) es evidente que hay $256!$ permutaciones posibles, y si de alguna manera asignamos a cada una de ellas un mensaje binario llegamos rápidamente a la conclusión de que podemos codificar con esto mensajes de hasta $\lfloor \log_2(256!) \rfloor = 1683$ bits (ver en bibliografía JOHN G. PROAKIS). En el caso concreto que nos ocupa, el programa *GifShuffle* es capaz de esteganografiar hasta 1675 bits de información secreta mediante la reordenación de una paleta de las características especificadas, por lo que es evidente que reserva para alguna función de control que el autor no especifica en la documentación 8 bits de los 1683 bits que puede albergar la paleta .

2. El *Snow* es bastante más sencillo. Básicamente codifica la información añadiendo espacios y/o tabuladores al final de las líneas de un documento de texto preexistente que hace de portadora. Es uno de los programas esteganográficos más sencillos y a la vez más conocidos. Tiene un pequeño problema: esta técnica puede detectarse muy fácilmente por software. Pero engañaría a la mayoría de observadores humanos ya que en general los espacios y los tabuladores del final de las líneas no tienen efectos visibles en los textos.

Básicamente la información secreta es codificada en el mensaje en bloques de 3 bits, añadiendo bloques de 0 a 7 espacios al final de las líneas. Los tabuladores se usan para separar bloques de espacios, ya que en líneas con poco texto pueden caber varios bloques de 3 bits (es decir, de entre 0 y 7 espacios) hasta llegar a las 80 columnas habituales de un fichero de texto. Además los tabuladores son necesarios cuando los 3 bits de información secreta son cero con lo cual hay que poner cero espacios: en ese caso dos tabuladores seguidos indicarían que ahí enmedio hay “cero espacios” y por tanto que hay codificados tres bits de información que valen cero.

Curiosamente el autor del programa indica que la codificación que parece más evidente usando espacios y tabuladores (escribir los bits uno a uno, añadiendo al final de las líneas del texto un espacio para indicar “cero” y un tabulador para indicar “uno”) no es la mejor, ya que aunque añade menos caracteres por bit (1 frente a 1.5 que añade su solución) en realidad requiere más columnas por bit (4.5 frente a 2.67) porque hay que tener en cuenta que el carácter “tabulador” tiene una representación visual que corre varias columnas (normalmente de 5 a 8) y el factor que más limita la codificación son las columnas vacías disponibles al final de cada línea hasta llegar a la columna 80 que es la última que suele utilizarse en textos normales.

5.1.8 MP3stego 1.1.15

Al igual que el formato gráfico JPEG, este formato de audio tan conocido utiliza compresión *lossy*. Sin embargo manipulando cuidadosamente la información es posible utilizarlo para propósitos esteganográficos, y esconder así información secreta en el *stream* de audio sin efectos relevantes sobre su calidad. Por otra parte el autor del programa sugiere que éste puede utilizarse como método de *watermarking* para audio en MP3, a la vez que reconoce que simplemente pasando el audio a otro formato o descomprimiendo+recomprimiendo el fichero (perdiendo en este caso algo de calidad) esta marca se borraría.

La compresión de audio MP3 es análoga a la compresión de imágenes JPEG, a pesar de tratarse de datos de naturaleza bien distinta. Hablando en general y utilizando terminología matemática, se desarrolla la señal en series de cosenos y luego se hace una aproximación de la imagen tomando los primeros coeficientes frecuenciales y descartando los últimos. Esa aproximación puede llegar a ser bastante buena a pesar de que supone descartar información, ocupa bastante menos que la señal original y además termina de comprimirse con Huffman o similares.

En el caso concreto de MP3, se pasan unos instantes de audio al dominio de la frecuencia mediante transformadas discretas y luego se lleva a cabo un proceso de cuantización controlado para que la información quepa en los bits disponibles (hay definidas unas tasas de transmisión que el audio MP3 debe cumplir, la más utilizada es la de 128Kbits/s que proporciona compresión 12:1) y para que además (a pesar de la distorsión de la cuantización) el resultado se ajuste en lo posible a los parámetros de calidad definidos por el llamado “modelo psico-acústico” que en principio garantiza que un oyente humano medio no detectará excesiva pérdida de calidad. Además, luego tenemos compresión de Huffman para reducir la mayoría de redundancias que puedan quedar.

Pues bien, la técnica utilizada por *MP3stego* es realmente retorcida y parece muy difícil de detectar: justo en mitad del complicado proceso de compresión de audio MP3 se obtiene como resultado una variable cuyo valor depende de la salida del bucle de cuantización, y la información secreta se esconde en la paridad de esa variable (i.e. en su bit menos significativo). Como no puede modificarse arbitrariamente el valor de esa variable después de obtenerlo porque resulta vital para la correcta descodificación de la información, lo que hace el *MP3stego* es alterar sutilmente la condición de salida del bucle de cuantización en función de la información secreta que queremos esteganografiar. Eso no produce efectos apreciables en el stream de audio porque la alteración es muy leve y además no se produce siempre, solo en instantes elegidos mediante un generador de números pseudoaleatorios basado en la función de *hash SHA-1*.

Dado que el formato de audio MP3 es muy popular, y dado que la técnica utilizada para esteganografiar la información es muy difícilmente detectable, este parece uno de los programas esteganográficos más interesantes de los actualmente disponibles.

5.1.9 GzipSteg (patch para gzip 1.2.4)

Este programa en realidad no es tal cosa, sino un “*patch*” para modificar uno de los compresores más utilizados actualmente en el mundo *Unix*: el *gzip*. Después de aplicar este parche al código fuente de la versión 1.2.4 y tras compilarlo, el ejecutable de *gzip* tendrá toda su funcionalidad anterior más una opción llamada “-s” o “-steg” que tiene un parámetro: el nombre del fichero que contiene (o que contendrá) la información secreta. Indicando esta opción en modo compresión

añadiremos el fichero secreto al fichero .gz resultante de una forma bastante difícil de detectar; y en modo descompresión además de extraer la información “visible” guardada en el fichero .gz también extraeremos la información secreta de ese fichero.

La técnica utilizada es relativamente simple: *gzip* usa compresión *LZ77* que en resumidas cuentas lo que hace es guardar pares de longitud+offset refiriéndose a ocurrencias anteriores del bloque de datos que está codificando actualmente. Si los datos de entrada contienen redundancias, este método las codifica de forma bastante eficiente. La longitud mínima tolerada por *gzip* es de 3 bytes, porque longitudes menores producirían pésima compresión o incluso expansión de los ficheros. Para esteganografiar información lo que hace el *GzipSteg* es codificarla en la longitud de los bloques: si la longitud del bloque es por lo menos 5, le resta 1 y pone el bit menos significativo a 1 ó a 0 en función del bit secreto a ocultar. De esta manera no queda una longitud mayor que la original (lo cual produciría una compresión errónea) ni llegamos en ningún caso a la longitud mínima de 3 bytes. Y la información secreta forma parte ya del bloque comprimido: ¡es el bit menos significativo de su longitud!

Por supuesto al acortar en ocasiones el tamaño del bloque codificado en un byte (al alterar el bit menos significativo de su longitud eso puede pasar) se produce un ligerísimo empeoramiento en la compresión, y además no podemos utilizar longitudes de bloque iguales a 4 lo cual también empeora ligeramente la compresión y podría parecer sospechoso en un análisis estadístico del fichero. Pero los ficheros generados mediante este método pueden descomprimirse con cualquier versión actual de *gzip*, y solamente las versiones “parcheadas” podrán extraer la información secreta. Como en muchos otros programas esteganográficos, es posible intentar extraer información secreta de cualquier fichero aunque no contenga esa información, porque ésta se codifica en la longitud de los bloques y eso es algo que tienen todos los ficheros .gz; eso hace que si la información secreta está encriptada y un atacante la extrae, tendrá difícil evaluar si se trata realmente de información... o no.

El punto débil de este método, que el autor no menciona es el siguiente: el *gzip 1.2.4* es uno de los programas más extendidos que hay en la actualidad, se usa muchísimo, tiene versiones para casi todas las plataformas existentes y los ficheros .gz son seguramente los más comunes en los servidores FTP de todo el mundo. Todas esas versiones de *gzip* son compatibles entre sí, y además es de esperar que en concreto la versión *1.2.4* para cualquier plataforma teniendo un fichero concreto como entrada dé siempre la misma salida, lo comprima de la misma forma. De hecho el *gzip* tiene 9 niveles de compresión diferentes (con diferentes *ratios* de compresión y diferentes tiempos) y algunas opciones que pueden modificar la salida, pero en general lo que queremos decir es que descomprimiendo un fichero con información esteganografiada y recomprimiéndolo (todo ello con *gzip 1.2.4*) la información secreta se perdería... y además el fichero recomprimido sería diferente del original, lo cual podría resultar sospechoso. Este problema ocurre al trabajar con una versión concreta de un programa y un formato concreto de fichero; esto no sucede por ejemplo con el formato *JPEG* porque hay miles de programas diferentes (o incluso dispositivos de hardware) que generan ficheros *JPEG* cada uno con su codec, y además el resultado varía muchísimo en función de la calidad especificada al comprimir.

5.1.10 Covert_Tcp

Este documento de CRAIG H. ROWLAND sobre canales de comunicación encubiertos en *TCP/IP* es realmente impresionante. Adjunta un poco de código fuente a modo de ejemplo (que funciona a pesar de ser un auténtico caos) y las ideas expuestas en el documento son brillantes. No son todas suyas: ha utilizado como fuentes principales la obra de COMER & STEVENS sobre *TCP/IP* (ver bibliografía), el “*Department Of Defense Trusted Computer System Evaluation Criteria*” y el artículo 6 del número 49 del *Phrack Magazine*. Sin embargo expone y resume las ideas con gran claridad.

La idea básica es que la pila de protocolos *TCP/IP* tiene unas características particulares que permiten la transmisión de información secreta ocultándola en paquetes de apariencia totalmente normal e inocente. Lo más ingenioso de los métodos propuestos en este documento/software es que no utilizan los campos más obvios para estos propósitos (los campos que en determinados

momentos se transmiten por homogeneidad pero no llevan información útil) sino que utiliza campos muy implicados en la comunicación que ningún *router* del camino osará modificar (cosa que podría pasar con los demás campos, que supuestamente son ignorados en recepción y que algunos routers alteran a su conveniencia).

Antes de ver las tres técnicas propuestas hay que saber algo de TCP/IP, lo básico. IP es un protocolo del nivel de red cuya misión es entregar paquetes de datos a la máquina correcta; los paquetes IP llevan la dirección del origen y del destino, que no son más que números de 32 bits; los paquetes IP son independientes unos de otros, llevan un identificador único, tienen el tamaño limitado a un máximo y van lógicamente sobre una red de conmutación de paquetes, y por tanto pueden llegar al destino desordenados, pueden llegar con retraso e incluso pueden perderse, y en todos esos casos IP no actuará; IP intenta hacer sus funciones lo mejor que puede, pero no arregla sus errores. Respecto a TCP, es un protocolo de la capa de transporte que funciona por tanto utilizando los servicios de IP y su función es proporcionar conexiones seguras sobre IP entre *aplicaciones* que pueden encontrarse en máquinas separadas; “seguras” significa aquí que TCP realiza todas las operaciones necesarias (solicitud de retransmisión, reordenación, etc) para que podamos tener -sobre esa caótica red de conmutación de paquetes donde lo único que se hace es *intentar* llevar los paquetes a su destino- unas conexiones que nos permitan intercambiar de forma fiable streams de datos entre dos aplicaciones en máquinas remotas, sin errores, sin duplicación ni desordenación de datos y con el byte como unidad básica. Dado que el byte es la unidad básica, el mecanismo de números de secuencia y de ACK se refiere a bytes individuales.

Las unidades de transmisión de TCP se llaman segmentos. Un segmento TCP se encapsula en un paquete IP para su transmisión sobre la red IP. El protocolo TCP es orientado a la conexión, y eso significa que para transmitir datos hay que establecer una conexión (lógica); y cuando ya no se necesita, hay que terminarla. El protocolo para establecer conexiones se llama “*Three Way Handshake*” y es relativamente simple: supongamos que A quiere conectarse con B, y veamos solo en lo que sucede cuando tiene éxito. Para empezar, A enviará a B un segmento TCP con el flag SYN activado y un número de secuencia inicial de 32 bits que llamaremos ISN y que es generado “aleatoriamente”. Luego, B le indicará a A que acepta la conexión enviándole un segmento TCP con los flags SYN y ACK activados, y el número de secuencia de *acknowledgement* que ahora valdrá ISN+1. Finalmente, la conexión quedará establecida cuando A le indique a B que conoce su conformidad mediante el envío de un segmento TCP con el flag ACK activado.

Después de esta brevísima introducción a TCP/IP estamos en condiciones de explicar los tres métodos propuestos por CRAIG H. ROWLAND para establecer comunicaciones encubiertas sobre esa pila de protocolos.

1. El identificador único de los paquetes IP.

Como hemos dicho, los paquetes IP son totalmente independientes entre sí: aunque varios paquetes IP lleven información sobre la misma comunicación, será TCP quien se encargue en el nivel superior de encadenarla de forma coherente. Además los paquetes llevan un número de identificación único de 16 bits. Este número es generado pseudoaleatoriamente y tiene como principal objetivo facilitar el reensamblado de los fragmentos de un paquete IP que haya sido dividido en varios trozos porque superaba la MTU (*Maximum Transmission Unit*) del medio de transmisión por donde ha pasado. Por ejemplo, los paquetes IP pueden ser mucho más grandes que 1500 bytes, que normalmente es la MTU de una red Ethernet. Si un paquete IP grande pasa por una red Ethernet (algo muy común) deberá ser fragmentado y luego reensamblado.

Bien, hemos dicho que este número de 16 bits es generado de forma pseudoaleatoria y que además es importante. Pues nada nos impide substituir esos 16 bits o parte de ellos por información secreta. Si el receptor sabe donde buscar, podrá leer sin demasiados problemas la información secreta, y eso es una forma simple de esteganografía en IP. Por supuesto 16 bits por paquete son muy pocos y además tenemos los problemas típicos de IP: los paquetes pueden llegar repetidos, desordenados, o incluso perderse, lo cual puede dar grandes problemas si la información secreta está cifrada con un algoritmo en el que el orden sea importante. Pero como mínimo los paquetes no llegarán con errores: hay un mecanismo de checksums bastante

eficiente que lo evitará. En cualquier caso para aplicaciones concretas y “muy secretas” de transmisión de información sencilla y principalmente dirigida a humanos, podría usarse. Puede resultar muy difícil de detectar, y sin embargo las dos técnicas siguientes son mejores.

2. El Número de Secuencia Inicial (ISN) del protocolo TCP.

Ya se ha explicado la utilidad de este número en la pequeña introducción a TCP/IP. En el primer paso del *Three Way Handshake* uno de los dos servidores (el que quiere iniciar la comunicación) manda un segmento TCP de tipo *SYN* con un número de secuencia llamado ISN. La aplicación esteganográfica resulta evidente: ese ISN pseudoaleatorio de 32 bits puede contener información esteganografiada, y el receptor podrá leerla sin demasiados problemas si tiene acceso directo al interface con IP.

En realidad eso requiere control de la pila de protocolos TCP/IP a un nivel más bajo que el habitual porque hay que manejar directamente el interface con IP para enviar segmentos TCP contruidos por nosotros mismos con propósitos muy concretos (es decir, en ciertos sistemas -como la mayoría de los Unix- solo alguien con privilegios de operador podrá hacerlo) pero tiene múltiples ventajas: si las peticiones de conexión van destinadas a puertos “inocentes” como el 80 para *HTTP* no despertarán demasiadas sospechas en la mayoría de *firewalls/proxies* que tengan que atravesar; además, ningún elemento del camino (a ningún nivel) modificará intencionadamente el ISN; y por si esto fuera poco, si luego el receptor no continua con el protocolo de conexión TCP y no contesta con el segmento *SYN,ACK*, típicamente (aunque no siempre) el intento de conexión no quedará registrado en ningún *log* ya que se considerará que al no haberse establecido la comunicación, no se ha transmitido ninguna información.

Este método, como los demás, requiere un esfuerzo importante del emisor y del receptor, que deben estar de acuerdo en los detalles más pequeños de la comunicación encubierta. También resulta difícil de detectar, pero no tanto como el siguiente método.

3. *Three Way Handshake* a tres bandas.

Esta técnica esteganográfica es sin duda una de las más retorcidas de las que hemos tratado aquí. Es una modificación de la anterior (transmite la información secreta en el número de ISN) pero utiliza un tercer servidor para enmascarar el origen de la transmisión y hace un paso más de los tres del *Three Way Handshake*, aunque sin establecer finalmente la conexión con lo cual normalmente nada queda registrado en los *logs*. La implicación de tres máquinas en un protocolo como el *Three Way Handshake*, diseñado para establecer conexiones fiables entre dos puntos es algo casi de ciencia ficción. Veamos como puede hacerse y como puede utilizarse además para transmitir información de forma muy encubierta.

Supongamos que A quiere mandar un mensaje secreto a C esteganografiado en segmentos TCP, y supongamos que además quiere hacerlo utilizando como intermediario un servidor muy ocupado llamado B. Esta técnica concreta fallaría en muchos entornos de red porque la mayoría de routers rechazarían este tráfico tan atípico (especialmente porque requiere la emisión de algunos paquetes con *IP spoofing* -“burla, falsificación”- de la dirección de origen), pero en ocasiones puede ser aplicable, y entonces sería muy seguro y difícil de detectar.

El proceso es el siguiente: A manda a B un segmento TCP de tipo *SYN* con el ISN conteniendo de alguna manera la información secreta y además, como dirección de origen no pone la suya sino que pone la de C (*IP spoofing*, requiere manipulación de las transmisiones a tan bajo nivel que en muchos sistemas operativos solo puede hacerse con privilegios especiales). Entonces, cuando B recibe el *SYN* debe contestar con segmentos con flags *SYN,ACK* si acepta la conexión, o con *SYN,RST* si la rechaza. La gracia está en que contestará con el número de secuencia de *acknowledgement* con un valor de $ISN+1$ y además esa contestación irá dirigida al que aparentemente solicita la conexión: ¡a C! Entonces C solo tendrá que leer ese número y restarle 1 para obtener el ISN original mandado por A, de donde podrá extraer la información secreta. Si no le contesta a B con un *ACK* no se terminará el *Three Way Handshake* y B dará por perdida la conexión tras un *timeout*, normalmente sin guardar nada en los *logs*. Si el servidor B es una máquina muy ocupada, este tráfico tan sospechoso pasará desapercibido entre otras muchas conexiones.

Estos tres métodos (sobre todo el tercero) si pueden aplicarse permiten comunicaciones lentas y con algunos problemas pero muy bien encubiertas y que pueden tener implicaciones de seguridad muy graves. Permiten, por ejemplo, acceder o incluso gestionar sistemas remotos (convenientemente preparados para estas técnicas, como es de suponer) atravesando *firewalls* sin problemas y sin despertar sospechas. Y además, permitirían desarrollar un nuevo tipo de troyanos que podrían robar información de un sistema y sacarla de él de forma muy “silenciosa” (difícil de detectar observando el tráfico y los *logs*), o incluso podrían permitir accesos desde el exterior en las mismas condiciones.

5.2 Software para otras plataformas

5.2.1 Digital Picture Envelop 1.0 (BPCS-Steganography)

Este programa demuestra una novedosa técnica esteganográfica que, a diferencia de las clásicas, permite esconder mucha información en una sola imagen. Trabaja principalmente con imágenes BMP (pero la técnica es extensible a cualquier tipo de bitmap), y gracias a BPCS una imagen de este tipo puede acoger prácticamente sin problemas un volumen de información de alrededor del 50% de su tamaño. Es decir, que si comprimimos la información secreta podemos llegar a esconder 1MB de texto en una imagen de 1MB, sin que prácticamente se note que está ahí.

El “secreto” de esta técnica consiste sencillamente en analizar qué bits de la imagen son “ruido” y qué bits de la imagen contienen información importante; esto es mucho más avanzado (y eficaz) que las técnicas clásicas que simplemente consideran que todos los bits bajos son ruido pero que los bits altos no deben tocarse bajo ningún concepto.

Los detalles del método (ver bibliografía) están en la web del autor, incluso con algunas demostraciones matemáticas y unos excelentes ejemplos visuales, pero a grandes rasgos consiste en descomponer la imagen en “*bitplanes*” que no son más que bitmaps monocromos; por ejemplo, una imagen con color de 24 bits se descompondría en 24 *bitplanes* monocromáticos: el primero tomando el bit más significativo del componente rojo (R)... y así sucesivamente hasta componer un *bitplane* con el bit menos significativo del componente azul (B), si suponemos ordenación RGB.

Entonces esos *bitplanes* son analizados para separar las regiones “ruidosas” de las que contienen información de forma. Las regiones “ruidosas” son prácticamente aleatorias y eso implica que los bits correspondientes a esas regiones pueden ser reemplazados por otros de similares características estadísticas sin afectar la calidad de la imagen. Como ya sabemos, los *bitplanes* correspondientes a los bits menos significativos suelen ser una gran región ruidosa; lo mejor del método es que además descubre y aprovecha regiones ruidosas en otros *bitplanes* correspondientes a bits más significativos.

En general los resultados de esta técnica son espectaculares por la calidad de las imágenes y por la gran cantidad de información que permite esteganografiar en una sola imagen. Realmente es una pena que una técnica tan inteligente y útil se implemente únicamente para Windows, en japonés, limitada a imágenes pequeñas (entre 64x64 y 512x512 pixels) y con “fecha de caducidad” (abril de 2000 por “motivos de seguridad”, según los autores).

5.2.2 Otras aplicaciones

En general con las aplicaciones para Unix que hemos comentado (y con esta última para Windows) cubrimos la mayoría de tipos de software esteganográfico disponible. Además, la mayoría de software disponible para otras plataformas (Windows, Mac, Amiga, cualquier cosa...) y prácticamente el resto de las aplicaciones para Unix que hemos podido encontrar implementan técnicas esteganográficas muy simples, basadas en la alteración de los bits bajos de algún tipo de muestras; prácticamente lo único que los distingue entre ellos en cuanto a funcionalidad son los algoritmos de compresión/criptación usados para procesar la información secreta antes de la esteganografía (si los usan) y los “extras” tales como borrado seguro de ficheros, entorno de edición “*anti-TEMPEST*”, etc.

6 Apéndice: esteganografiar las herramientas esteganográficas

A primera vista parece un problema “recursivo”, pero veremos que no es así y que además es un problema real que en ocasiones pide a gritos una solución práctica.

6.1 El problema

La esteganografía nos ayuda a ocultar la existencia de información secreta (y posiblemente cifrada) escondiéndola dentro de otras informaciones menos interesantes que difícilmente llamarán la atención de un hipotético “enemigo”. Pero resulta que independientemente de si hacemos esto para almacenar una información secreta o para transmitirla, es evidente que lo hacemos motivados por la sospecha (o la certeza) de la existencia de ese “enemigo” y su interés por los secretos que queremos ocultarle.

El mundo real es bastante complicado y las situaciones que pueden motivar el uso de la esteganografía son innumerables, pero parece evidente que en ocasiones no solo queremos esconder los datos sino también las herramientas que usamos para protegerlos. La esteganografía es mucho más útil cuando el “enemigo” ignora que la usamos e ignora como la usamos, porque la seguridad ofrecida por la esteganografía se basa en eso (aunque no de forma única). Además, en muchas de las situaciones que pueden motivar el uso de la esteganografía puede ser tan negativo el descubrimiento por parte del “enemigo” de las herramientas esteganográficas (o criptográficas), como el descubrimiento de la propia información que ocultamos.

Por ejemplo, si decidimos encriptar y luego guardar esteganográficamente nuestros secretos comerciales en nuestro álbum de fotos digitales familiares, y por algún motivo nuestro disco duro cae en manos del “enemigo” y éste descubre herramientas criptográficas y esteganográficas junto a ese abundante álbum de fotos, tendremos más problemas que si encuentra el álbum solamente. De la misma manera, si intentamos sacar información de un país “hostil” y la tenemos esteganografiada en unos ficheros de audio que aparentan ser inocentes notas de voz en nuestro PDA último modelo, tendremos problemas serios en el caso de que al intentar salir del país nos registren el PDA y encuentren además de las notas un software que permita esteganografiar documentos en ficheros de audio. No sabemos de ningún software esteganográfico para PDA pero sin duda puede hacerse, solo es cuestión de tiempo hasta que aparezca alguno. Y de todas formas, no era más que un ejemplo.

6.2 Posibles soluciones

Ahora que hemos visto como puede ser necesario (en ocasiones) ocultar las herramientas criptográficas o esteganográficas, veamos si es posible hacerlo y cómo puede hacerse.

Una primera aproximación al problema sugiere utilizar medios físicamente distintos para separar los datos del software. En general esa es una buena política de seguridad frente a accidentes de todo tipo, pero en este caso además evita que si un atacante consigue acceder a los datos sospeche que hay información esteganografiada en alguna parte. Por ejemplo, podríamos tener el software esteganográfico en un disquete que guardaríamos en un lugar seguro, o podríamos tener el software en el disco duro y el álbum de fotos en un CD-RW (regrabable) que también guardaríamos lejos del equipo informático. Sin embargo esto tan sencillo no siempre es posible, ni es lo mejor.

Otra aproximación simplista consistiría en borrar las herramientas esteganográficas una vez utilizadas, sin dejar ningún rastro. Esto en principio es posible porque la mayoría de herramientas esteganográficas están disponibles en la Web en todo momento, y si no hace falta acceder con frecuencia a los datos no supone un problema muy grande bajarse el software cuando es necesario. Evidentemente es un sistema engorroso y además los sistemas operativos actuales hacen muy difícil borrar todo rastro de una aplicación. El borrado seguro y efectivo de datos y de aplicaciones requeriría un trabajo aparte, pero es algo realmente *muy* difícil, en todas las plataformas existentes. De todas formas BRUCHE SCHNEIER hace una pequeña introducción al tema en su famoso *Applied Cryptography*.

Finalmente tenemos las soluciones más o menos buenas que (como siempre) no son las más sencillas. Existen muchas otras soluciones, pero la que nos parece más interesante se basa en una técnica muy utilizada en la industria del software y a la que (milagrosamente) hemos encontrado una utilidad práctica: hablamos de los “*easter eggs*” (literalmente, Huevos de Pascua). Esos “Huevos de Pascua” no son más que pequeñas “sorpresas” que los programadores incluyen en sus programas. Normalmente se activan tras una tediosa secuencia de pulsaciones de teclas, introducción de comandos y clicks del ratón sobre lugares insospechados, y suelen consistir en la presentación de los autores del software de una forma más o menos espectacular. Por ejemplo, en el Excel 95 tras una complicada secuencia de acciones sobre la hoja de cálculo se pone en marcha un juego de lucha en un laberinto 3D, tipo Doom; si el jugador es suficientemente habil puede encontrar varias habitaciones (muy bien escondidas) en las que podrá leer los nombres de los principales autores del programa. Análogamente, el Word 97 lleva un juego de Pinball y el Excel 97 una especie de simulador de vuelo. Y ya hemos hablado de los “Huevos de Pascua” que podemos encontrar en la calculadora HP-49 de Hewlett Packard, aunque antes hemos utilizado directamente la palabra “esteganografía” para referirnos a ellos. En el fondo aunque la intención sea diferente la técnica usada es la misma.

Pues bien, aunque parezca increíble esas retorcidas técnicas de programación son potencialmente útiles: pueden servir para esconder software esteganográfico dentro de otro software que aparentemente realiza otra función totalmente inocente. Si puede esconderse un simulador de vuelo en una hoja de cálculo, puede esconderse una herramienta esteganográfica en un software cualquiera. Que sepamos, nadie lo ha hecho hasta ahora, ni lo ha propuesto, pero es algo perfectamente posible.

Pero tiene el mismo problema de siempre: hay que mantenerlo en secreto para que sea completamente efectivo. Si por ejemplo Microsoft decide vender el “StegOffice 2000”, con programas esteganográficos a modo de “Huevos de Pascua”, puede que venda muchos como curiosidad (o puramente por marketing, como hace siempre) pero esa herramienta esteganográfica oculta (suponiendo incluso que funcione) no será de mucha utilidad en muchos casos porque su existencia será de dominio público: quien utilizara “StegOffice 2000” estaría confesando que tiene secretos que ocultar, y que utiliza la esteganografía para ello, y eso no siempre es conveniente.

El siguiente paso (bastante “esotérico”, como diría BRUCE SCHENIER) consiste en aplicar el concepto de “*plausible deniability*” para esconder software esteganográfico dentro de otro software de manera que uno pueda negar (convincientemente) que no está ahí; aunque esté. Eso ya es demasiado para nosotros, aunque bien pensado nunca se es suficientemente paranoico...

7 Ejemplos para ilustrar este documento

Este documento trata sobre todo de técnicas esteganográficas modernas, de la “era digital”. Hemos hablado de esconder información en textos ASCII, en imágenes digitalizadas, en sonidos digitalizados, en dispositivos, etc. Todo esto está muy bien y ya lo hemos tratado con la profundidad que queriammos. Pero en este documento falta algo muy importante: ejemplos reales de esteganografía. En realidad, tiene unos pocos gráficos y además puramente decorativos...

Todo esto tiene una explicación: en un documento impreso no es posible mostrar imágenes que contengan información esteganografiada, porque la imagen original y la alterada se verían prácticamente iguales y ambas serían severamente modificadas respecto de su forma digital; tampoco podemos mostrar ejemplos con audio, y además sobre un papel el texto ASCII no existe como tal y los espacios y tabuladores al final de las líneas no tienen ningún sentido.

Para paliar un poco esta falta de ejemplos gráficos, que en muchas ocasiones son los que ayudan realmente a entender las cosas, hemos procurado explicar las técnicas utilizadas de forma sencilla y comprensible, tan extensamente como nos ha parecido necesario, lo cual nos ha permitido a nosotros mismos entender la esteganografía mucho mejor que antes de empezar este trabajo. Además, en nuestra cuenta *dtsxt08* del laboratorio 30 (*Sun*) del edificio *Anselm Turmeda* hemos guardado en el directorio */home/docencia/dtsxt08/treball/* varios ejemplos de fotografías tomadas con cámara digital (*Fuji MX-2900-Z*) en dos versiones: las originales (*moixa.jpg*, *lab30.jpg*) y

las alteradas que contienen información esteganografiada (moixa_s.jpg, lab30_s.jpg). En esas imágenes hemos escondido respectivamente el código fuente integro del *DES* (una versión *beta* del *des.c*, procedente de la práctica) y un documento en *HTML* sobre *Ciberguerra*. Ambos documentos fueron comprimidos con *gzip* antes de esteganografiarlos. La genuina procedencia de las imágenes (lab30.jpg sobre todo) es indiscutible...

Además, en ese mismo directorio tenemos el código fuente del *Jsteg* (es decir, el *Jpeg-v4* ya parcheado) y los dos ejecutables resultantes (*cjpeg*, *djpeg*), ya compilados para *Sun*, para hacer las pruebas oportunas. Compilarlos no fué fácil: requiere editar el *Makefile* y algunos ficheros de configuración para adaptar algunas cosas a la particular arquitectura *Sun*. Los ejecutables utilizan por defecto entrada y salida estándar, y mostrarán todas las opciones al recibir el parámetro “-help”. De todas formas, y a modo de resumen, diremos que para extraer en el fichero “*X.gz*” la información que se halla esteganografiada en “*Y_s.jpg*” simplemente hay que usar la siguiente instrucción: `'cat Y_s.jpg | djpeg -steg X.gz > /dev/null'` con lo cual descomprimos la imagen (volcando el resultado al dispositivo nulo, ya que no interesa) y además extraemos la información “secreta”.

8 Bibliografía

Antes de empezar con la bibliografía, queremos comentar la entrada más importante: la página Web de ERIC MILBRANDT, en <http://members.tripod.com/steganography/stego.html>. Es una recopilación excelente (creemos que difícil de superar) con buen material sobre esteganografía en la Web y realmente nos ha servido para encontrar gran parte de la información que hemos usado aquí, de manera que cualquier referencia que se nos olvide poner estará probablemente en esa página Web.

A parte de eso, hemos utilizado principalmente las siguientes fuentes de información, de las que indicaremos: el nombre de los autores, el título de la obra (que bien puede ser software, en cuyo caso nos estaremos refiriendo también a su documentación), la URL si es aplicable y, ocasionalmente, en que aspecto de la obra nos hemos centrado si ésta es muy amplia. Hay que tener en cuenta que debido a la escasez de documentos más o menos didácticos sobre esteganografía muchas veces hemos extraído la información de la documentación y el código fuente del software existente.

- ROSS ANDERSON, ROGER NEEDHAM & ADI SHAMIR. *The Steganographic File System*. (Hemos usado sobre todo algunos argumentos en favor de la esteganografía, y la idea de “*plausible deniability*”)
- BRUCE SCHNEIER. *Applied Cryptography*, 2nd. ed. (Nos ha servido la introducción que hace a la esteganografía, al borrado seguro de datos, y todo lo relacionado con criptografía y seguridad en general.)
- SHU LIN, DANIEL J. COSTELLO. *Error control Coding: Fundamentals and Applications*. (Hemos buscado más detalles de los códigos de GOLAY)
- DOUGLAS E. COMER, DAVID L. STEVENS. *Internetworking With TCP/IP*. (Sobre todo el Volumen I, con detalles de los paquetes IP y los segmentos TCP).
- JOHN G. PROAKIS. *Digital Communications*. (Especialmente el Capítulo 3).
- (?) *U.S. Patent Number 5,613,004 "Steganographic Method and Device"*.
- EIJI KAWAGUCHI. *BPCS-Steganography, Digital Picture Envelope*.
<http://www.know.comp.kyutech.ac.jp/BPCS>
- DEREK UPHAM. *Jsteg*.
<ftp://ftp.funet.fi/pub/crypt/steganography/jpeg-v4.tar.gz>
<ftp://ftp.funet.fi/pub/crypt/steganography/jpeg-jsteg-v4.diff.gz>

- ULRICH KUEHN. *Stext*.
<http://???>.
- STEVEN HUGG. *StegParty*.
<http://cometbusters.com/hugg/projects/stegparty.html>
- STEFAN HETZL. *StegHide*.
<http://www2.crosswinds.net/~shetzl/steghide/index.html>
- NIELS PROVOS. *OutGuess*.
<http://www.outguess.org>
- MARKUS KUHN. *StirMark*.
<http://www.cl.cam.ac.uk/~mgk25/stirmark/>
- MATTHEW KWAN. *GifShuffle* y *Snow*.
<http://www.darkside.com.au/gifshuffle/>
<http://www.darkside.com.au/snow/index.html>
- FABIEN A.P. PETITCOLAS. *MP3Stego*.
<http://www.cl.cam.ac.uk/~fapp2/steganography/mp3stego>
- CRAIG H. ROWLAND. *Covert Channels in the TCP/IP Protocol Suite*.
<http://www.psionic.com/papers/covert.tcp.tar.gz>
- ANDREW BROWN. *Hiding things in gzip files*. Publicado en el grupo de news *sci.crypt* dia 1994.04.28.
- TRAD. DA5ID. *Cypherpunk Manifesto*. WWW.
- TARIQ JAMIL. *Steganography*. IEEE Potentials (febrero/marzo 1999).
- CHU-HSING LIN & TIEN-CHI. *Ejemplo de esteganografía en textos*. Computers & Security, num. 6, 1998.