STEGANOGRAFIA

DANIELE ASSERETO

L'ARTE DI TRASMETTERE CON OCCULTE SCRITTURE I VOLERI DEL PROPRIO ANIMO A CHI È LONTANO

27 APRILE 2001

INTRODUZIONE

Cenni Storici

Il termine Steganografia deriva dal greco, e più precisamente dalle parole *stèganos*, che significa nascosto, e *gràfein*, che significa scrivere. Furono infatti i Greci tra i primi ad usare dei rudimentali sistemi di steganografia, incidendo del testo nascosto su delle tavolette di legno ricoperte di cera. Per mandare un messaggio segreto una persona avrebbe dovuto quindi grattare via la cera da una tavoletta, incidervi sotto il messaggio, e successivamente ricoprire la tavoletta con della cera per farla apparire nuova.

La *Steganografia* non va confusa con la *Crittografia*, in quanto quest'ultima tecnica punta la sua sicurezza sul fatto che tutti possono leggere un messaggio criptato, ma in teoria solo il destinatario può decriptarlo e quindi riuscire a leggerlo. La debolezza della crittografia è proprio il sistema usato per criptare il messaggio: una volta abbattuto quello, si ha libero accesso a tutti i segreti trasmessi. La steganografia invece parte dal presupposto che nessuno dovrebbe sapere che esiste un messaggio segreto nascosto, e quindi risulta impossibile trovarlo. L'importante è quindi escogitare un metodo sicuro per poter nascondere il proprio segreto.

I Greci solevano usare anche un'altra tecnica, come narra lo storico Erodoto: rasavano a zero la testa del messaggero, vi tatuavano sopra il testo segreto, aspettavano che gli ricrescessero i capelli e successivamente lo inviavano al destinatario dandogli anche un messaggio fittizio per non destare sospetti. Una volta giunto, il messaggero si rasava il capo e lo mostrava al destinatario.

Nell'antica Cina invece si dipingeva il messaggio su striscioline di seta finissima, che venivano appallottolate e coperte di cera. Le palline erano quindi inghiottite dal messaggero.

Risulta evidente che tutti i metodi descritti finora potevano essere facilmente scoperti ad un esame molto accurato. La longevità della steganografia dimostra comunque ch'essa garantisce una certa sicurezza, ma il suo punto debole è palese: se il latore del messaggio è attentamente perquisito, è probabile che il testo sia scoperto; in tal caso, il nemico può farne l'uso che crede. In altre parole, la segretezza è perduta nel momento

stesso dell'intercettazione. In tal caso, è inevitabile che molti messaggi siano trovati.

Giovanni Tritemio

Il primo vero passo in avanti fu compiuto dall'abate Giovanni Tritemio (1462-1516) con i suoi trattati *Steganographia* e *Clavis Steganographiae,* nei quali venivano gettate le basi per un modello di Steganografia usata ancora oggi. Ma in cosa consisteva di preciso l'idea di Tritemio?

Supponiamo per un istante di voler mandare un messaggio ad un nostro amico Tizio, avvertendolo di non fidarsi di un certo Caio. Abbiamo però paura che Caio legga tutta la nostra corrispondenza e quindi, usando la Steganografia, scriviamo il seguente testo: "*Nelle ore notturne feroci illusioni di antichi riti tramandati in dimenticate isole ci assalgono, ivi ora…*" ecc. ecc… Tizio, per leggere il messaggio originale che noi gli volevamo mandare, non dovrà far altro che leggere tutte le iniziali delle parole e comporre il testo nascosto: "*Non fidarti di Caio…*".

Questo è l'esempio più semplice di tutti gli schemi proposti da Tritemio, che elaborò 40 sistemi principali e 10 sotto-sistemi secondari, sfruttando non solo varie combinazioni di acronimi, ma anche usando dei dischi rotanti (come quello riportato qui a fianco) basati sulla sostituzione mono-alfabetica di Cesare. In questa cifratura il posto di ogni lettera del messaggio è preso dalla lettera che si trova ad una distanza di *x* posti nell'alfabeto ordinario, dove *x*, nel caso dell'alfabeto completo di 26 lettere, è un numero compreso tra 1 e 25.

Il messaggio "*Mio zio è andato a Zurigo non per un semplice incontro notturno di karate, quindi domani si farà il solito giretto nel centro storico. Dovrebbe mandarmi un kimono per sabato, e allora…*", prendendo una parola si e una no, nasconde la stringa, a prima vista illeggibile, "*zazpsnkdfsnsmksa*". Usando però la trasposizione del disco riportato qui sopra, che quindi tramuta le '*a'* in '*o*', le '*b*' in '*p*', le '*c*' in '*q*' ecc…, ecco che riusciamo ad ottenere il solito "*nonfidartidicaio*".

L'idea fondamentale di Tritemio era quindi quella di nascondere un testo segreto dentro un messaggio che funzionasse come copertura, senza quindi ricorrere a brutali sistemi fisici come ad esempio la rasatura di capelli ma sfruttando invece abili artifizi matematici e letterari. A meno di non sapere il sistema usato per nascondere il testo, era dunque praticamente impossibile riuscire a estrarre il significato reale del messaggio.

Unico inconveniente: il mittente ed il destinatario dovevano avere entrambi il libro di Tritemio per poter conoscere il sistema steganografico usato…

STEGANOGRAFIA MODERNA

Terminologia

Oggi per Steganografia si intende l'arte e la scienza di nascondere un'informazione in modo tale che la sua presenza non possa essere determinata.

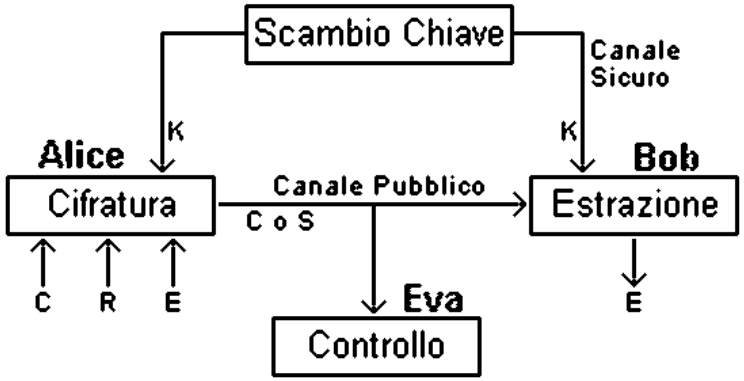
A causa dell'interesse crescente circa la protezione della proprietà su Internet e anche nell'eventualità che le tecnologie crittografiche siano dichiarate fuorilegge, l'interesse nelle tecniche per nascondere informazioni è aumentato notevolmente negli ultimi anni.

Oggi si possono quindi distinguere due principali diramazioni nell'attuale scenario steganografico:

1. la protezione contro una semplice individuazione del messaggio da parte di un *avversario passivo.*
2. l'abilità di nascondere un messaggio in modo tale che nemmeno un *avversario attivo* possa rimuoverlo.

La Steganografia con un *avversario passivo* può essere illustrata con il "*Problema dei Prigionieri*" di Simmons: Alice e Bob sono in prigione ed intendono progettare un piano di fuga. Tutte le loro comunicazioni sono osservate da un avversario (la guardia Eva), che ostacolerà il loro piano trasferendoli in una prigione ad alta sicurezza non appena individuerà

anche il minimo messaggio nascosto. Alice e Bob riusciranno nel loro intento se Alice riuscirà a mandare messaggi a Bob senza che Eva si accorga di nulla e possa insospettirsi.



Lo schema qui sopra mostra come funziona lo scambio di messaggi con un avversario passivo. **E** è il *testo segreto* che Alice e Bob si vogliono scambiare, **C** è il *testo di copertura*, **R** è un'eventuale *variabile random* di Alice, **K** è la *chiave segreta* condivisa in modo sicuro da Alice e Bob, **S** è il *messaggio steganografato* che Bob riceve.

Quando Alice vuole mandare un messaggio a Bob, sa che deve passare sotto il controllo della guardia, e quindi nasconde il messaggio E dentro di C, eventualmente utilizzando R, ottenendo così il risultato S. Eva vede che Alice sta comunicando con Bob, ma non sa se quello che sta trasmettendo sia C o S, anche perchè non ha modo dato S di tornare ad E. Bob invece, quando riceve un messaggio di Alice, prova a decifrarlo con la chiave K, e scopre se vi era nascosto un testo segreto. In questo modo Alice e Bob riescono ad aggirare la sicurezza di Eva, comunicando tranquillamente grazie alla chiave K. Va sottolineato il fatto che il messaggio S dovrebbe essere insospettabile, e avere un suo senso più o meno compiuto.

Nascondere un messaggio in modo tale che nemmeno un *avversario attivo* possa rimuoverlo è un problema leggermente differente, e più complesso. Se nello schema sopra Eva non si limita a guardare i messaggi che si scambiano Alice e Bob, ma ne altera anche alcune parti senza modificarne il significato, ecco che Bob non sarà più in grado di risalire al

testo segreto originale di Alice. Eva però non può sapere dove sia nascosto il segreto, quindi può solo cambiare una parte del messaggio dove pensa che vi sia nascosto qualcosa. È proprio su questa insicurezza di Eva che deve basarsi Alice, quando nasconde il testo segreto.

La terminologia che rispetta gli schemi di attacco e rottura della steganografia è simile alla terminologia crittografica; vi sono comunque alcune sostanziali differenze. Mentre un crittanalista applica la crittanalisi per cercare di decodificare o rompere messaggi criptati, lo *steganalista* è colui che applica la steganalisi cercando di determinare l'esistenza di eventuali informazioni nascoste. In crittografia si poteva partire dal confronto di porzioni di testo in chiaro e parti di testo cifrato. Con la steganografia invece, il confronto può essere fatto tra il testo di copertura, il messaggio steganografato ed eventuali porzioni di testo segreto. Il messaggio finale steganografato inoltre può anche essere criptato, e in questo caso una volta scoperto esso deve ancora venir decriptato.

Quali sono gli attacchi possibili alla steganografia? Diciamo che possiamo distinguere i seguenti casi:

1. Solo steganografico
2. Copertura conosciuta
3. Testo conosciuto
4. Steganografia usata
5. Messaggio scelto

Nell'attacco *solo steganografico* abbiamo a disposizione solo il messaggio steganografato S per le nostre analisi. Questo è probabilmente il caso peggiore che ci si possa presentare. Se invece siamo a conoscenza sia di C sia di S, possiamo parlare di attacco con *copertura conosciuta*. La steganalisi può sfruttare anche il *testo conosciuto* quando il testo nascosto viene rivelato qualche tempo dopo, e quindi l'attaccante può cercare di analizzare il messaggio S per attacchi futuri. Anche conoscendo il testo segreto E può essere comunque molto difficile risalire al metodo steganografico usato, e quindi questo caso può essere paragonato al primo caso solo steganografico. L'attacco a *steganografia usata* è quando lo steganalista conosce l'algoritmo usato per nascondere il testo. Infine l'attacco del *messaggio scelto* si ha quando lo steganalista genera il

messaggio steganografato S usando qualche algoritmo e partendo da un testo segreto conosciuto. Lo scopo di questo attacco è determinare parti corrispondenti nel messaggio steganografato S che possano rivelare quale algoritmo o programma è stato usato.

Il testo segreto può essere nascosto in vari tipi di file, e i più usati al giorno d'oggi sono le immagini e i file audio o video.

Limiti Teorici

È possibile definire uno schema che assicuri una copertura assoluta, allo stesso modo del cifrario a blocco monouso in crittografia?

Supponiamo che Alice usi un segnale video digitale non compresso come *copertura* **C**, e che codifichi il *testo segreto* **E** nelle frequenze veramente basse. Per esempio, il *k*-esimo bit di E diventa il bit meno importante di uno dei pixel del *k*-esimo fotogramma del video, e la scelta del pixel è specificata dalla *k*-esima parola di un cifrario a blocco monouso condiviso con Bob. Intuitivamente ci aspettiamo che un attacco sia impossibile: il testo segreto sarà completamente nascosto nel rumore intrinseco del video. C'è qualche modo per dimostrarlo?

Dobbiamo prima chiederci come deve essere una prova di copertura. Una definizione accettabile di *stegosistema sicuro* potrebbe essere un sistema nel quale nessun intruso possa accorgersi di differenze tra il testo di copertura C e il testo steganografato S contenente l'informazione nascosta E, a meno che non conosca la chiave K. Come per la crittografia, possiamo supporre che l'intruso sia una macchina di Turing probabilistica polinomiale nel caso necessitassimo di sicurezza computazionale, e possiamo assumere che possa esaminare tutte le possibili chiavi nel caso necessitassimo di sicurezza assoluta.

Alla fine, l'intruso riuscirà a vedere il messaggio nascosto, e quindi lo stegosistema sicuro deve generare un numero sufficiente di messaggi nascosti sensati da un qualsiasi stego-testo S dato, e il numero di questi messaggi non deve essere distinguibile in alcun modo dallo stego-testo e da un qualsiasi testo di copertura.

Quello che rende la steganografia più complessa dei problemi di segretezza o autenticazione è che dipendiamo tantissimo dal nostro modello di testo di copertura.

Chi lavora in Teoria dell'Informazione spesso assume che qualsiasi sorgente di informazione può essere compressa in modo tale che non vi sia nessuna ridondanza. Questa assunzione può essere molto utile per trovare limiti asintotici o risultati di capacità, ma ha anche un effetto abbastanza curioso se applicata alla steganografia.

Supponiamo di aver creato un *sistema perfetto* di codifica in una scatola nera fisica che può sia comprimere sia decomprimere dati di un tipo particolare (audio, video, o qualsiasi altra cosa). Per compressione perfettamente efficiente si intende che gli oggetti compressi saranno condensati in una serie di stringhe di bit di lunghezza uguale. In questo modo Alice potrà prendere un qualsiasi testo segreto che voglia nascondere e farlo passare dentro il compressore. Il risultato sarà una registrazione audio, video o altro, perfettamente accettabile.

Questo esempio non è una prova rigorosa. È possibile, per esempio, che un dispositivo possa decomprimere una stringa di bit casuale di lunghezza n in un tipo particolare di oggetto con probabilità polinomiale di $1/n$. Questo risultato può essere sufficiente ad andare avanti in molti casi di teoria delle informazioni, ma non va bene per l'esempio sopra. Infatti mostra che molte intuizioni classiche di teoria delle informazioni possono servire a poco per descrivere il comportamento di sistemi steganografico. Abbiamo quindi che la steganografia pratica può funzionare solo se la compressione è inefficiente. Se fosse efficiente sarebbe impossibile risalire al testo nascosto.

Se Alice e Bob usano un blocco a cifratura monouso condiviso per decidere quale bit del testo di copertura sarà sostituito da un bit di testo segreto, possiamo considerare il blocco come a un *canale di selezione*. Se Eva è veramente molto potente, può provare tutte le possibili cifrature (inclusa quella corretta), e quindi il numero di cifrature con le quali nascondere il testo segreto deve essere abbastanza grande da non far sospettare che Alice stia mandando del testo steganografato piuttosto che un testo in chiaro.

Può essere utile a questo punto richiamare l'esempio del libro usato per la cifratura. Alice e Bob condividono un libro e cifrano un messaggio come una serie di puntatori a parole. In questo modo il numero "78216" può significare pagina 78, paragrafo 2 e 16-esima parola. I libri con i codici devono essere scelti in modo che Eva non sappia quali possano essere, e bisogna far attenzione a non riutilizzare la medesima parola del

libro. La cifratura tramite un libro è un esempio di canale di selezione. In questo caso si parte dal libro e successivamente viene generato il testo segreto, mentre in uno stegosistema si parte dal testo da nascondere e si genera successivamente lo stego-testo; ma nel caso in cui il canale di selezione sia veramente random (un blocco a cifratura monouso), allora sono la stessa cosa, perché un messaggio qualunque può essere nascosto in un testo di copertura qualsiasi o sufficientemente lungo.

Un libro ripetitivo avrà quindi una bassa capacità, e sarà possibile usare una piccola percentuale delle sue parole per costruire degli attacchi seri. Allo stesso modo, se il testo di copertura usato nello stegosistema gode di statistiche insolite (ad esempio un numero non equilibrato di uni e zeri) allora la sua capacita steganografica sarà bassa, in quanto solo una piccola porzione dei possibili messaggi cifrati potrà sembrare abbastanza casuale.

L'approccio coi canali di selezione può suggerire un'idea migliore.

Continuiamo ad usare un blocco a cifratura monouso non per selezionare un pixel ma una serie di essi, e nascondiamo il bit di testo segreto in base alla loro parità. In questo modo l'informazione può essere nascosta cambiando uno qualsiasi dei pixel che possono essere cambiati senza che creino troppo rumore.

Dal punto di vista della teoria dell'informazione, se ciascun bit del testo di copertura e "1" con probabilità 0.6, allora la probabilità che un paio di bit abbiano parità 1 è 0.52; se consideriamo tre bit, la parità è 1 con probabilità 0.504, e così via. Allora codificando ciascuno bit nascosto in base alla parità di k bit dello stego-testo, possiamo ridurre l'effetto che il processo steganografico ha sulle statistiche dello stego-testo sotto qualsiasi soglia arbitraria; e poiché il miglioramento è geometrico, non dovremo in pratica incrementare k di molto. È come se utilizzando più bit nel canale di selezione (ad esempio incrementando il valore di k) aumentassimo anche i bit che possiamo nascondere nel nostro testo di copertura.

Nascondere Informazioni nelle Immagini

Cominciamo a vedere qualche metodo per nascondere informazioni nei file comuni, come ad esempio le immagini.

Per un computer, un'*immagine* è un array di numeri che rappresenta le intensità di luce di vari punti (i pixel). Questi pixel specificano le caratteristiche di un'immagine; abbiamo ad esempio che le foto più

comuni sono di 640 X 480 pixel e 256 colori (cioè 8 bit per pixel). Le immagini digitali sono tipicamente strutturate in file da 24 o 8 bit; un'immagine da 24 bit ha più spazio a disposizione per nascondervi informazioni.

Tutte le variazioni di colore per i pixel derivano dai tre colori primari: rosso (R), verde (G) e blu (B). Ciascun colore primario è rappresentato da un byte, quindi le immagini da 24 bit usano 3 byte per pixel per rappresentare il valore di un colore. Questi 3 byte possono essere rappresentati tramite valori esadecimali, decimali o binari. In molte pagine web, ad esempio, il colore dello sfondo è rappresentato da un numero di sei cifre esadecimale, formato quindi da tre coppie che rappresentano i colori primari RGB. Uno sfondo bianco avrà quindi il valore FFFFFF: 100% di rosso (FF), 100% di verde (FF) e 100% di blu (FF). In valore decimale saranno 255, 255 e 255, mentre in binario 11111111, 11111111 e 11111111.

Per nascondere dei dati in un'immagine avremo quindi bisogno di due file: innanzitutto una foto di copertura (che chiameremo **C** in base allo schema precedente con avversario passivo) che possa sembrare innocente a prima vista e che nasconderà le informazioni, e poi il messaggio che vogliamo nascondere (che sarà **E**). Questo messaggio potrà essere del testo in chiaro, un testo cifrato, un'altra immagine, o comunque qualsiasi altra informazione codificata in bit. Una volta combinati insieme, i due file genereranno la stego-immagine **S**.

Le informazioni possono quindi essere nascoste utilizzando metodi diversi. Tra gli approcci più comuni per farlo vanno ricordati:

1. LSB: Inserimento nel Bit meno Significativo.
2. Algoritmi e Trasformazioni.

L'*inserimento nel bit meno significativo* (*LSB*) è il più comune e semplice tra gli approcci per nascondere informazioni di qualsiasi tipo in un'immagine. Sfortunatamente, è molto vulnerabile in caso di una anche minima manipolazione dell'immagine da parte di terzi.

Usando ad esempio un'immagine da 24 bit, possiamo immagazzinare 3 bit in ogni pixel, e all'occhio umano la stego-immagine risultante apparirà identica all'immagine di partenza. Volendo ad esempio nascondere la lettera A (in binario 10000011) nei seguenti 3 pixel:

(00100111 11101001 11001000)
(00100111 11001000 11101001)
(11001000 00100111 11101001)

otterremo il seguente risultato:

(00100111 1110100**0** 11001000)
(0010011**0** 11001000 1110100**0**)
(11001000 00100111 11101001)

I bit in grassetto sono i soli tre che sono stati cambiati. In generale, quindi, LSB richiede che in media vengano cambiati la metà dei bit che vengono ritoccati. Volendo si può nascondere l'immagine utilizzando anche il secondo bit meno importante, e ancora l'occhio umano non noterebbe la differenza.

Se però Eva ritoccasse leggermente l'immagine, ad esempio schiarendola anche di pochissimo, l'intera informazione nascosta andrebbe persa irrimediabilmente.

Una piccola variante potrebbe essere il metodo chiamato "Patchwork", che altera le statistiche dell'immagine di copertura. Dapprima vengono selezionate coppie di regioni dell'immagine usando un generatore di numeri casuali. Successivamente l'intensità dei pixel di una regione è incrementata di una valore costante mentre i pixel dell'altra regione sono in corrispondenza decrementati dello stesso valore. La modifica è tipicamente piccola e non percettibile, ma non per forza limitata ai bit meno significativi.

Con determinati *algoritmi e trasformazioni* invece vengono sfruttate, ad esempio, le caratteristiche della compressione JPEG (e quindi le trasformate di Fourier e le trasformate discrete del coseno) per includere il messaggio nascosto direttamente nell'immagine prima di crearla. Queste tecniche nascondono il testo in aree più significative dell'immagine manipolando ad esempio proprietà come la luminosità, e sono quindi più robuste rispetto ad LSB.

Lo scopo dei metodi basati su trasformazioni è quello di rendere l'informazione nascosta parte integrante dell'immagine, in modo tale che cercando di eliminarla o di disabilitarla si distrugga la stego-immagine stessa, e quindi sia inutilizzabile.

Sono stati presentati svariati algoritmi per nascondere dati, tra cui va segnalata una tecnica in cui si utilizza il messaggio per modulare un segnale portante, che viene poi combinato con la stego-immagine in sezioni di blocchi non sovrapposti. Il messaggio verrà quindi estratto da una correlazione incrociata tra la stego-immagine e l'elemento originale ricostruito.

Esistono poi metodi statistici che calcolano le proprietà di "dithering" di un'immagine: vengono scelti opportuni pixel in base alle loro caratteristiche ed il valore in quei punti viene sostituito con l'informazione nascosta. Un ulteriore algoritmo utilizza la compressione di immagini frattali: un bit di informazione è nascosto nella stego-immagine trasformando un blocco simile nell'approssimazione di un altro; i dati vengono decodificati usando una chiave visuale che specifica la posizione delle regioni che contengono il messaggio; sfortunatamente, la quantità di dati che possono essere nascosti usando questo metodo è piccola e suscettibile a errori di bit; e come se non bastasse, la ricerca di blocchi simili nell'operazione di codifica ed il processo di comparazione nella decodifica sono entrambe operazioni computazionalmente molto costose.

Software: S-Tools

Questo programma sviluppato da Andrew Brown permette di nascondere anche più documenti nello stesso oggetto, che puo' essere un'immagine (JPG, GIF, BMP) o un file sonoro (WAV). I file da nascondere vengono compressi e successivamente nascosti includendo i loro nomi. Il tutto viene infine criptato (con MD5) usando una password scelta dall'utente. Sarebbe forse troppo semplice se i dati venissero nascosti come una sequenza lineare di bit, e allora S-Tools crea un numero casuale generato sempre dalla password e lo usa per scegliere la posizione del bit successivo dell'oggetto di copertura da usare.

Se per esempio in un file con 100 bit disponibili per nascondere informazioni se ne vogliono nascondere 10, S-Tools non sceglie i bit da 0 a 9 che potrebbero essere facilmente identificati da un eventuale nemico, ma potrebbe scegliere i bit 63, 32, 89, 2, 53, 21, 35, 44, 99, 80, o altri 10, che vengono determinati univocamente dalla password.

Ma come vengono nascosti in pratica i dati nei due tipi di file? Vediamo in principio i file sonori. Questi sono, per loro natura, stime non precise del

corretto valore dell'onda sonora in un particolare momento del tempo. Ad esempio i file sonori WAV in Windows sono immagazzinati usando 8 o 16 bit (che vengono poi eventualmente convertiti da una scheda sonora). Un file da 8 bit implica che i valori possono essere in un range da 0 a 255. Il range dei 16 bit varia invece da 0 a 65535. Tutto quello che S-Tools fa è distribuire la sequenza di bit che corrispondono al file da nascondere nei bit meno significativi del file sonoro. Ad esempio, ammettiamo che siano presenti i seguenti otto byte di informazione da qualche parte:

132 134 137 141 121 101 74 38

che in binario diventano:

10000100 10000110 10001001 10001101
01111001 01100101 01001010 00100110

Supponendo di voler nascondere il byte 11010101 (cioè 213) in questa sequenza, il programma semplicemente sostituisce i bit meno significativi di ogni byte con il corrispondente bit estratto dal byte che si vuole nascondere. In questo modo la sequenza originaria è diventata:

133 135 136 141 120 101 74 39

che in binario è:

10000101 10000111 10001000 10001101
01111000 01100101 01001010 00100111

Come si può quindi vedere, i valori del file sonoro sono cambiati, al massimo, di un valore per ciascun byte. Questa differenza non sarà assolutamente percepibile all'orecchio umano.

Nel caso delle immagini il ragionamento è lo stesso del metodo LSB descritto sopra, con qualche particolare accorgimento nel caso delle immagini GIF dato che sono una formato compresso a 8 bit. La palette dei 255 colori alla fine del processo risulterà leggermente diversa da quella originaria, in quanto i valori dei pixel sono stati cambiati nel corso dell'operazione.

Il programma S-Tools è oramai giunto alla versione 4, rilasciata dal suo autore nel 1996. È freeware, cioè può essere copiato e distribuito gratuitamente con il rispetto delle leggi del proprio paese.

Watermark e Fingerprint

Vediamo adesso invece un esempio pratico di cosa nascondere in un'immagine. Prendendo immagini da 24 bit, possiamo imprimere dei "marchi" di due tipi, e più precisamente avremo:

1. Watermark
2. Fingerprint

I *Watermark* sono un classico esempio dell'utilizzo di informazioni nascoste in un'immagine. Ma in che cosa consiste, in pratica, un watermark?

Diciamo che con il diffondersi dell'utilizzo di Internet, gli autori di documenti digitali (come possono essere ad esempio le immagini) hanno il problema di impedire che qualcuno si impossessi delle loro opere spacciandole per proprie. Molti autori però vogliono poter distribuire i propri lavori in modo sicuro, sapendo cioè che nessuno potrà mai contestare la paternità dell'opera. I watermark digitali quindi provvedono ad aggiungere delle informazioni dentro il documento in questione in modo tale che, nel caso ne venisse fatta una copia, il legittimo proprietario possa essere univocamente determinato.

Sempre nel caso delle immagini, un modo semplice per creare un watermark può essere quello di creare una seconda immagine che non conterrà altro che il nome del proprietario ripetuto molte volte. Questa tecnica non viene usata quindi per nascondere un messaggio generico, ma piuttosto per occultare informazioni che vogliamo siano presenti nel documento ma che non siano visibili ad un primo impatto.

In definitiva quindi si può definire il watermark come uno o più marchi di copyright nascosti nel contenuto del messaggio.

I *Fingerprint* invece sono dei marchi distinti che vengono inseriti in diverse copie della stessa immagine per distribuirla a persone diverse. L'effetto è una specie di numero seriale: fa sì che il proprietario possa

identificare le persone a cui ha distribuito il suo prodotto nel caso esse lo distribuiscano "gratuitamente" a terzi.

Vediamo adesso alcune debolezze dei marchi e quindi alcuni modi che potrebbero essere utilizzati per renderli inutilizzabili.

Una pratica comune è quella di distribuire il watermark per tutta l'immagine, impedendo in questo modo eventuali attacchi basati sul taglio di una sezione dell'immagine di partenza. Più piccolo è il watermark e più è facile che venga manipolato.

Assumiamo quindi che la stego-immagine **S** sia composta da una parte visibile **C** e una invisibile **E**, e quindi S = C + E. Qualsiasi manipolazione a S porterà quindi ad una modifica di C e di E. Gli attacchi ai watermark non saranno per forza dei tentativi che puntano alla rimozione, ma anche più semplicemente a renderli illeggibili. L'attaccante potrà quindi cercare di aggiungere ulteriori marchi all'immagine cercando di coprire quelli esistenti. Ovviamente, se dovessero risultare presenti più di un marchio sulla stessa immagine, avrà la precedenza il più vecchio, essendo stato inserito prima di tutti gli altri.

Per evitare attacchi di questo tipo bisogna ricordarsi ad esempio di inserire watermark dai colori molto contrastanti, in modo da essere sicuri che non possano diventare illeggibili se sottoposti a filtri particolari o a trasformate di vario tipo. Anche la dimensione è importante: non deve essere troppo grande per evitare che se ne possa perdere una parte, e deve essere ripetuta per tutta l'estensione dell'immagine per evitare, come abbiamo già detto sopra, che si possa asportare quella parte di immagine che contiene il marchio.

File System Steganografico

Si può pensare di usare la steganografia anche per creare qualcosa di completamente diverso e più particolare. Si può pensare di mettere in piedi un *file system* basato sulla steganografia. Lo si può vedere come un meccanismo di immagazzinamento dei dati che ha lo scopo di dare all'utente un livello molto alto di protezione contro la rivelazione stessa dei suoi contenuti. Verrà permesso l'accesso ad un file a tutti quegli utenti che conoscono il suo nome e la relativa password; ma un attaccante che non sia in possesso dell'informazione corretta non potrà nemmeno

immaginare la sua esistenza, anche avendo un completo accesso a tutto l'hardware ed il software. Sono possibili due costruzioni diverse.

Una prima idea potrebbe basarsi sul volere che l'attaccante non abbia nessuna conoscenza del nome del file e della sua password, e in questo modo non possa ottenere nessuna informazione sul sapere se il file sia presente nel sistema a meno che non conosca una parte del suo contenuto o possa tentare tutte le password possibili. Per semplicità nell'esposizione, si può assumere che la password **P** sia formata da k bit. L'idea principale è di avere un certo numero di *file di copertura* (cover files) nel sistema, che vengono inizializzati con valori random, e successivamente si nascondono i file dell'utente con delle operazioni di *or esclusivo* con un sottoinsieme dei file di copertura. Il sottoinsieme viene scelto in base alla password. Successivamente il risultato ottenuto andrà "xorato" con uno dei $\mathbf{C}_j$ e sostituito ad esso.

Per dare un esempio concreto, si supponga che tutti i file abbiano la stessa lunghezza e che ci siano k file di copertura nel sistema con cui iniziare, chiamati $\mathbf{C}_0,...,\mathbf{C}_{k-1}$. Supponiamo anche che il primo file dell'utente sia **F** e che la sua password sia **P**. Allora vengono selezionati quei $\mathbf{C}_j$ tali che il j-esimo bit di P abbia valore 1 e si applica loro l'operazione di XOR bit per bit; il risultato viene ancora "xorato" con il file fornito dall'utente e con un $\mathbf{C}_j$. Il risultato è che il file dell'utente F è ora l'or esclusivo di un sottoinsieme dei C_j selezionati dai bit non nulli di P. In altre parole, abbiamo:

$$F = \bigoplus_{P_j=1} C_j$$

Un utente potrà quindi nascondere un numero di file pari alle combinazioni possibili dei file di copertura. Aggiungere successivamente altri file richiede di risolvere insiemi di equazioni lineari per decidere quale combinazione dei C_j vada alterata.

Una proprietà importante di questo sistema è che si avrà una gerarchia di accessi lineari (cioè un utente che memorizza un file con un dato livello di sicurezza conosce le password di tutti i file memorizzati ai livelli inferiori) e quindi i file possono essere aggiunti in modo naturale senza disturbare i dati già nascosti in precedenza.

L'algebra lineare mostra un possibile attacco al sistema: se la dimensione della password è k ed il nemico fosse a conoscenza di più di k

bit del testo in chiaro, allora dopo aver ottenuto tutti i file casuali dal computer potrebbe scrivere k equazioni lineari nel k numero di bit sconosciuti della chiave. Questo è la ragione per cui si è supposto in partenza che il nemico non conosca niente del testo in chiaro. Se questa assunzione vale, allora l'intera struttura appare vulnerabile solo ad un nemico che conosca la password.

In un sistema pratico però si potrebbe voler gestire dei files di lunghezze diverse, ed anche avere la protezione sulle directory oltre che sui singoli files. Tutto questo può essere accomodato nel sistema descritto qui sopra immagazzinando ogni directory in uno dei file F_j la cui lunghezza potrebbe essere fissata a (per esempio) 10 MB; 100 di queste directory potranno essere facilmente immesse in un tipico PC con 3.5 GB di hard-disk, lasciando comunque abbastanza spazio libero per il sistema operativo e per software e dati non protetti.

In ogni caso questa costruzione comporta due problemi, uno maggiore ed uno minore. Quello minore è che potrebbe non essere sempre conveniente avere una gerarchia stretta di accessi lineari, mentre quello maggiore è che ci sarebbe una notevole penalizzazione nelle prestazioni: leggere o scrivere un file comporterebbe accedere ad una grande quantità di dati, soprattutto nel caso in cui si stia modificando l'intera directory.

A questo punto ci si potrebbe chiedere: perchè non viene semplicemente riempito l'intero hard-disk con *bit casuali*, e poi si scrive ciascun blocco di file ad un indirizzo assoluto del disco scelto in base a qualche processo pseudocasuale, come ad esempio criptando la locazione del blocco nel file usando come chiave una *funzione hash* del nome del file e della password della directory? Il blocco potrebbe inoltre essere criptato usando una chiave derivata usando un processo simile e così, assumendo di avere un blocco cifrato che il nemico non possa distinguere da una permutazione casuale, la presenza o l'assenza di un blocco a qualsiasi locazione non potrebbe essere distinguibile.

Il problema principale di un simile approccio è che, in base al teorema del compleanno di Teoria delle Probabilità, si comincerebbe ad avere "collisioni" (cioè riscrittura di aree di dati già utilizzate) subito dopo aver scritto poco più di $\sqrt{N}$ blocchi, dove **N** è il numero di blocchi presenti sul disco. Esiste comunque una soluzione: scrivere il blocco in più di una locazione. Se, per esempio, ogni blocco venisse scritto in due locazioni scelte pseudocasualmente, allora entrambi questi blocchi verrebbero

sovrascritti da un terzo solo quando il numero dei blocchi totali sia approssimativamente nell'ordine $\mathbf{O}(N^{2/3})$. Si dovrebbe quindi aggiungere ridondanza ad ogni blocco prima di criptarlo, in modo che il sistema possa identificare quando un blocco è stato sovrascritto e cercare l'altra locazione in cui esso era stato salvato. Continuando su questo ragionamento, si avrebbe che se venissero scritte 10 locazioni uguali, allora prima che tutte le locazioni contenenti un blocco siano accidentalmente sovrascritte potremmo scrivere $\mathbf{O}(N^{9/10})$ blocchi, e così via.

Sono già presenti sul mercato dei pacchetti di software che realizzano un tentativo di file system steganografico, tra cui va ricordato Steganos, che oltre a fare quanto voluto mette a disposizione anche le normali tecniche steganografiche già descritte per S-Tools. Purtroppo non essendo freeware non è possibile sapere a quale costruzione si siano ispirati i programmatori per implementare il file system.

Altri Metodi

Esistono poi altre tecniche che sono state pensate allo scopo di nascondere informazioni e riuscire a trasmetterle a terzi. I *codici per la correzione d'errore* per esempio, che normalmente sono utili per risolvere i problemi che possono insorgere dai sistemi di trasmissione, possono essere utilizzati per nascondere informazioni, ad esempio trasmettendo degli errori in piccole porzioni di bit. Quando i bit vengono letti, i codici per la correzione d'errore eliminano automaticamente l'errore ed un lettore casuale non sarà mai in grado di scoprire che il file originale conteneva degli errori; per ripristinare i dati corretti basta quindi utilizzare il codice per la correzione d'errore e salvare il messaggio che era stato nascosto in precedenza come errore.

Una seconda tecnica può essere scrivere un programma che produca *testo imitativo* che appaia statisticamente simile al testo originale. Si possono trasformare in modo automatico piccoli, innocui bit di dati abbellendoli con dettagli profondi e fioriti, fino a che il significato venga ad assomigliare a qualcosa di completamente diverso. In questo modo, è possibile nascondere i dati. L'algoritmo di compressione di Huffman riesce ad approssimare la distribuzione statistica del testo ed a convertirla

in stenografia digitale. Facendolo funzionare al contrario si possono prendere i dati normali ed adattarli a schemi elaborati.

Il tutto può essere poi notevolmente complicato generando *grammatiche* complesse che imitino il linguaggio umano ed in cui sia possibile nascondere dei dati. Il risultato così ottenuto può essere spedito quindi a vari newsgroup di Internet; individuare le differenze tra questo risultato ed i vari commenti che si trovano normalmente sui newsgroup è veramente difficile.

Un esempio pratico può essere il seguente: si desidera iniziare una conversazione con un amico all'altro capo del mondo. Invece di inviare i bit delle parole direttamente all'altra persona, il programma per la conversazione li codifica in un testo innocuo, come ad esempio la discussione di una partita di calcio della sera prima. Il programma può anche essere abbastanza intelligente da accedere ad un database on-line per ottenere una vera tabella dei risultati, in modo che la discussione sembri vera. Quando l'altra persona risponde, il suo programma di conversazione codifica la risposta con la stessa grammatica. La vera conversazione potrebbe riguardare argomenti molto privati, ma apparirebbe come una qualsiasi chiacchierata sul calcio per chiunque stia spiando dal circuito.

L'idea può essere vista come un perfezionamento automatizzato delle idee di Tritemio, senza però il vincolo degli acronimi o comunque delle singole lettere.

Conclusioni

Esistono infine ulteriori programmi che permettono di nascondere informazioni in filmati di vario tipo ed in formati diversi di file audio, come ad esempio il diffusissimo MP3.

Attualmente si sta lavorando al concetto di Steganografia di Internet, e cioè alla possibilità di nascondere informazioni ed eventualmente essere in grado di recuperarle dagli header dei pacchetti TCP/IP o da altre trasmissioni di rete. Non bisogna dimenticare che anche se la steganografia e la crittografia sono discipline indipendenti, possono essere impiegate per alterare ed occultare il medesimo testo, garantendo un livello di sicurezza molto più alto.

E se qualcuno obiettasse che ci potrebbe essere qualcosa di sbagliato in tutti questi algoritmi, e che se un criminale potesse nascondere le informazioni in modi così efficaci la polizia diventerebbe meno efficiente,

non si può fare altro che rispondere con un pizzico di buon senso: i disonesti non obbediranno mai ad una legge che imponga loro di non utilizzare alcuna forma di steganografia o altro. Il problema fondamentale è se l'onesto cittadino possa avere a disposizione gli strumenti che gli consentano di proteggere la propria privacy. La crittografia e la steganografia danno agli individui questo potere.

PREGHIERA DI TRITEMIO

O Lettore, chiunque tu sia, ti prego e scongiuro, nel Santo nome di Iahwè, per il Sangue di Nostro Signore Gesù Cristo, Salvatore delle nostre anime, che il segreto di questa scrittura non sia rivelato a persone ignoranti perfide spudorate lascive e maligne, che la usino indegnamente. Se fosse scoperto il senso riposto, molti mali ne deriverebbero: adulteri, fornicazioni, cospirazioni, omicidi e tradimenti.

L'uomo probo ed onesto può e deve servirsene per esprimere, in modo sicuro ed insospettabile, il suo segreto volere, quando lo ritiene necessario per l'utile pubblico o privato; invece il disonesto ed impudico, accecato da passione carnale, se ne servirebbe solo per comunicare con l'amante, travolta dalla stessa infame passione, e fissare i luoghi, la data ed i termini dei loro convegni, attraverso parole innocenti, pudiche e sante.

Non esisterà più rispetto e lealtà tra i coniugi se la donna, impudica, potrà facilmente, senza destare alcun sospetto, leggere attraverso sante e nobilissime parole, lodate dallo stesso marito, il turpe capriccio del suo drudo e l'amante, con linguaggio simulato, potrà comunicare alla donna, ciò che brama, al cospetto del marito, anzi tramite lui stesso.

Questa scrittura crittografica, divulgata poi in campo politico, sconvolgerebbe ogni ordine: lo Stato perderebbe ogni garanzia e credito; tutto: documenti, carte, scritture, le parole stesse sarebbero gravate da sospetto.

Nessuno crederebbe più agli scritti anche se nobili e santi né avrebbe più fiducia alcuna nelle lettere, le parole, anche le più innocue e sante, sarebbero sospettate e tutti diverrebbero timidi e perplessi nel conversare e nello scrivere.

Se qualcuno mi obiettasse, dicendomi: "se volevi che codesta tua scrittura rimanesse segreta, perché l'hai scritta?", ecco la risposta: "non ho

né voluto, né potuto tenerla celata ad un ottimo principe, che, ad ogni occasione, potrà liberare i suoi devoti (anche io mi annovero tra questi) da molti pericoli e potrà far conoscere la sua volontà, in modo segreto, senza il timore che il suo nemico possa decifrarla".

Referenze

Libri:

- *"Steganographia"*, Giovanni Tritemio, 1606.
- *"Clavis Steganographiae"*, Giovanni Tritemio, 1606.
- *"Cryptomenytices",* Gustavo Seleno, 1624.
- *"Crittografia Invisibile"*, Peter Wayner, 1997.
- *"Codici e Segreti"*, Simon Singh, 1999.

Articoli:

- *"The Steganographic File System"*, Ross Anderson & Roger Needham & Adi Shamir, 1997.
- *"Exploring Steganography: Seeing the Unseen"*, Neil Johnson & Sushil Jajodia, 1998.
- *"Steganalysis of Images Created Using Current Steganography Software"*, Neil Johnson & Sushil Jajodia, 1998.
- *"An Information-Theoretic Model for Steganography"*, Christian Cachin, 1998.
- *"On The Limits of Steganography"*, Ross Anderson & Fabien Petitcolas, 1998.
- *"Attacks on Steganographic Systems"*, Andreas Westfeld & Andreas Pfitzmann, 1998.
- *"Steganography in a Video Conferencing System"*, Andreas Westfeld & Gritta Wolf, 1998.
- *"Reliable Blind Information Hiding for Images"*, Lisa Marvel & Charles Boncelet & Charles Retter, 1998.
- *"An Introduction to Watermark Recovery from Images"*, Neil Johnson, 1999.

Siti web:

- http://utenti.tripod.it/tritemio/
- http://members.tripod.com/steganography/stego.html
- http://www.cl.cam.ac.uk/~fapp2/steganography/
- http://www.jjtc.com/Steganography/
- http://www.steganos.com/