Steganografia

La steganografia è una pratica nota fin dall'antichità che ha lo scopo di rendere sicura la comunicazione. A differenza della crittografia, il cui obiettivo è rendere la comunicazione incomprensibile per chiunque ad eccezione di chi è autorizzato, la steganografia cerca di rendere invisibile l'atto comunicativo in sé.
L'origine del termine risale all'antica Grecia ed è traducibile come "scrittura nascosta": già all'epoca le tecniche per occultare i messaggi erano diverse.
Vista la diffusione di tavolette di cera per la scrittura, un semplice ma efficace metodo steganografico consisteva nell'incidere il messaggio da trasmettere direttamente sul fondo legnoso della tavoletta di cera, prima che questa venisse effettivamente applicata, per poi scrivere sopra la cera un messaggio diversivo. Lo storico Erodoto racconta di come il re persiano Demarato riuscì a trasmettere un allarme riguardo un imminente attacco alla Grecia proprio in questo modo. Il tiranno di Mileto Istièo, per istigare una rivolta contro i persiani fece tatuare un messaggio sulla testa del suo servo più fidato, dopo averlo fatto rasare, così che il messaggio risultasse del tutto invisibile dopo che i capelli furono ricresciuti.
Altri esempi noti di "steganografia fisica" riguardano la scrittura di messaggi con inchiostri invisibili, messaggi in codice morse su fili di lana usati poi per realizzare indumenti, messaggi minuscoli coperti poi dai francobolli, eccetera...

Steganografia digitale

La steganografia si è spostata da mezzi fisici al digitale ed è utilizzata sia per la trasmissione invisibile di messaggi, sia per il watermarking dei contenuti.

Steganografia mediante immagini o suoni

La forma di steganografia più comune consiste nel nascondere informazioni all'interno di file immagine o audio. Questa tecnica si basa sulla teoria che modifiche ai bit meno significativi delle immagini o dei file audio sono impercettibili per l'uomo e difficilmente individuabili tramite analisi statistiche a causa del rumore di fondo intrinseco dei tipi di file usati come contenitori.
Immaginiamo di avere a disposizione un'immagine bitmap, in cui il colore di ogni pixel è espresso come somma di altri tre colori, detti anche canali: rosso verde e blu. Questa codifica è detta RGB e la quantità di ciascun colore può variare nello spazio [0,255], ciò significa che per ogni pixel sono necessari 3 byte per specificare i valori di colore. Se però variamo ciascuno dei bit meno significativi delle tre componenti, possiamo inserire 3 bit di informazione senza modificare l'aspetto del contenitore in modo sensibile. Queste variazioni possono tranquillamente essere confuse con il rumore delle foto, se non sono eseguite sistematicamente per ogni pixel, oppure con le distorsioni derivate dai sistemi di compressione audio. Un discorso equivalente può essere applicabile anche ai video, che possono essere validi contenitori, soprattutto se si considera che dalle dimensioni del contenitore dipende la quantità di contenuto che si può nascondere all'interno.

Esempio:

Testo da nascondere: vediamoci alle 12
Codifica ASCII:
01110110011001010110010001101001011000010110110101101111011000110110100100100000011000010110110001101100011001010010000001100001011011000110110001100101001000000011000100110010
Caratteri in chiaro: 17
Bit in codifica ASCII: 17*8 = 136
Pixel necessari per la codifica: 136/3 = 46
Una icona di una pagina web 14px*14px può contenere il nostro messaggio:

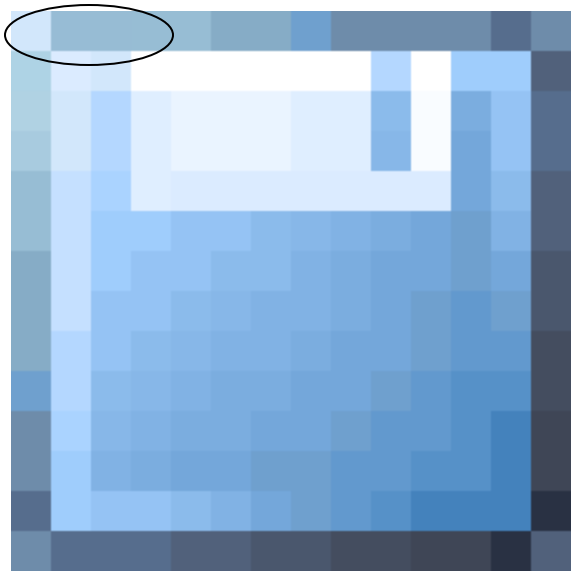


Figura 1 Immagine originale

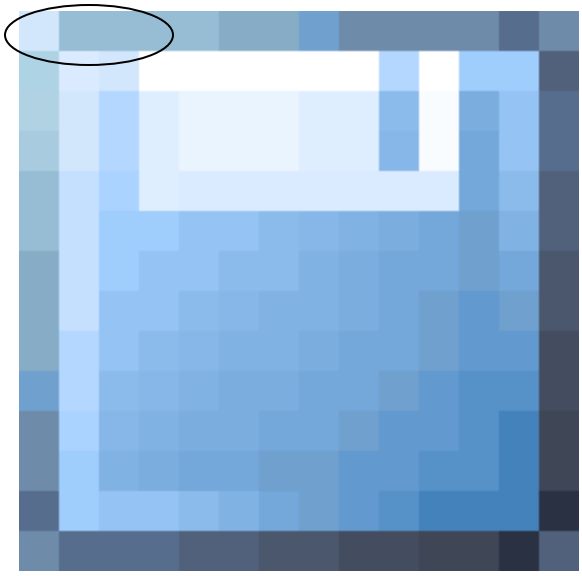


Figura 2 Immagine Modificata

Nell'esempio, un'icona di dimensioni 14x14, ingrandita 20 volte, mostra come non ci siano grandi differenze tra le due immagini[*]. Di seguito viene riportata la tabella delle modifiche effettuate:

Pixel	Valori RGB originali			Colore	Valori RGB modificati			Colore	Valore
00	11010010	11100111	11111011		11010010	11100111	11111011		011
01	10010111	10111101	11010100		10010111	1011110**0**	1101010**1**		101
02	10010111	10111101	11010100		10010111	1011110**0**	11010100		100
03	10010111	10111101	11010100		10010111	10111101	11010100		110
…									

Alcune variazioni dell'algoritmo prevedono di modificare più di un solo bit per ciascun canale, così da poter sfruttare immagini di dimensioni minori per trasmettere grandi quantità di informazioni, mentre per immagini indicizzate si può usare un sistema ancora migliore.
Le immagini indicizzate includono una porzione di dati (detta palette) che definisce tutti i colori utilizzati all'interno dell'immagine: questo formato è utile soprattutto per immagini realizzate

[*] Solo i primi 4 pixel dell'immagine sono stati modificati, manualmente, tramite un editor di immagini.

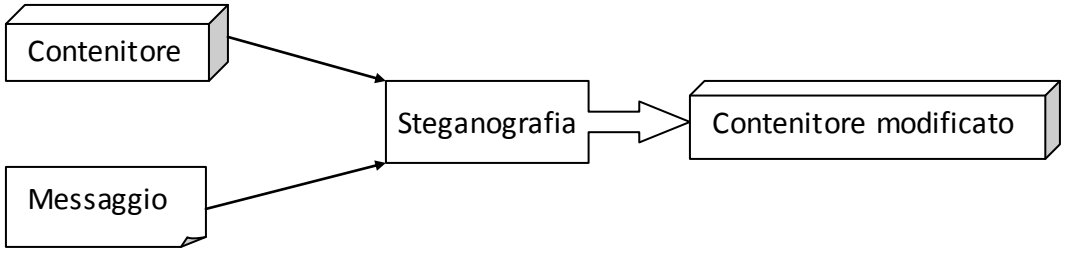
interamente al computer che fanno uso di pochi colori. La palette viene creata indicizzando i colori utilizzati nell'immagine, in questo modo anziché specificare i valori dei tre canali RGB per ogni pixel, questi si specificano solo una volta nella palette. Ciascun pixel che compone l'immagine farà riferimento all'indice di un qualche colore nella palette. I colori indicizzati possono essere disposti in qualunque ordine, senza che questo modifichi in alcun modo l'aspetto dell'immagine, ad esempio si possono riordinare in modo che gli ultimi bit di ogni colore formino un messaggio nascosto, oppure si possono aggiungere colori non utilizzati all'interno dell'immagine al solo scopo di contenere dati particolari, oppure ancora si possono ordinare i colori in modo che la loro distanza di Hamming componga il messaggio, eccetera.

Tecniche steganografiche e livelli di protezione[1]

Scomporre un messaggio e nasconderlo all'interno di un contenitore può non essere sufficiente per mantenere la segretezza delle informazioni, soprattutto se un attaccante è in grado di ottenere una copia del contenitore non modificato: mettendo a confronto le due versioni del contenitore, le differenze verrebbero immediatamente alla luce, rivelando il contenuto segreto.
Per questa ragione esistono diversi livelli di protezione associati a diverse tecniche di steganografia, ed in ogni caso si preferisce crittografare il messaggio prima di nasconderlo nel contenitore.

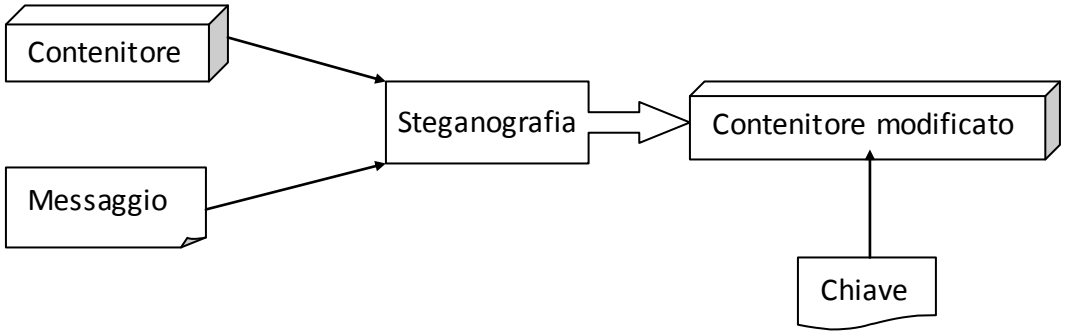
Livello 1

Il primo, più semplice, corrisponde a quello che è stato analizzato nell'esempio precedente, in cui il messaggio è semplicemente camuffato all'interno del contenitore.



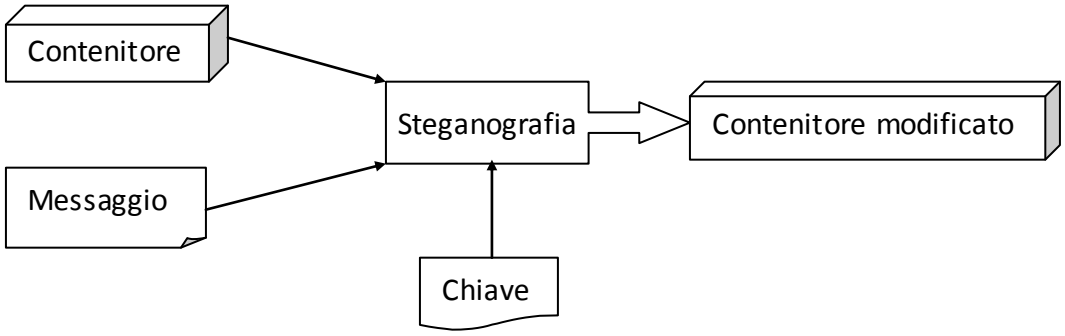
Livello 2

Il secondo livello di protezione (così come tutti i successivi) prevede l'utilizzo di una chiave, concordata tra mittente e destinatario, inserita (in chiaro o crittografata) all'interno del contenitore modificato. Questo sistema è utilizzato ad esempio per aggiungere una firma all'interno del file come prova di copyright o per dimostrarne l'origine.



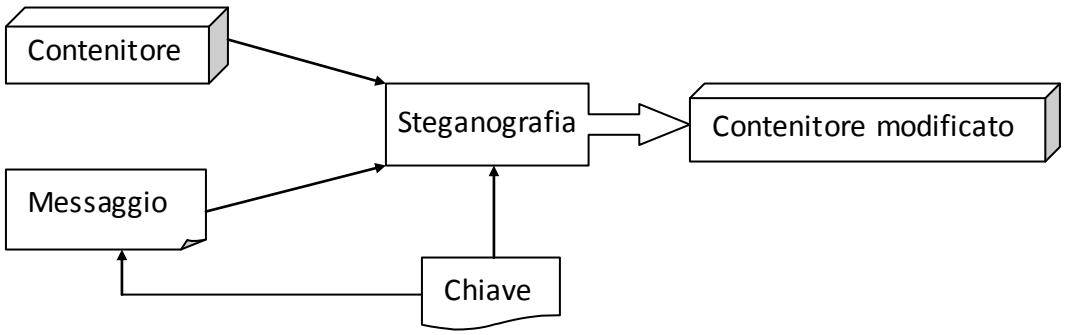
Livello 3

Il terzo livello di protezione influenza la distribuzione del messaggio all'interno del contenitore in base alla chiave tramite una particolare funzione di distribuzione. La funzione scompone il messaggio in parti e stabilisce quanto deve essere grande ciascuna di esse, e quanto devono distare due porzioni consecutive di informazioni all'interno del contenitore.



Livello 4

Il quarto ed ultimo livello prevede la presenza di due diverse funzioni di distribuzione, una responsabile della selezione di parti del messaggio, e la seconda responsabile della selezione della posizione all'interno del contenitore.



Altri esempi

Un altro tipo di steganografia digitale è la blog-steganography, in cui i messaggi appaiono come commenti crittografati ad articoli di blog e/o come messaggi all'interno di forum, e la chiave che permette di recuperare il contenuto nascosto consiste nell'insieme ordinato dei blog in cui il messaggio è stato separato.
Secondo la stampa, tecniche steganografiche simili a questa sono state utilizzate da terroristi per nascondere testo all'interno di avatar nei forum ed all'interno di immagini di oggetti all'asta su eBay.com.
Informazioni crittografate possono essere steganografate all'interno di contenitori composti da altri dati crittografati. Ad esempio il software opensource TrueCrypt permette di creare due archivi crittografati separati all'interno di un unico file: così facendo è possibile nascondere dati confidenziali insieme ad altri dati di scarsa importanza sfruttando due diverse password di accesso. Nel caso si venisse obbligati a rivelare la chiave dell'archivio, si può comunicare quella che decritta l'archivio falso, mantenendo i dati protetti.

Steganografia di rete

Le tecniche steganografiche possono anche essere impiegate in ambiente di rete per trasmettere messaggi nascosti. Solitamente per l'invio di messaggi tramite rete vengono modificati i dati contenuti in alcuni pacchetti di un determinato protocollo, oppure si interviene sugli intervalli di tempo tra due trasmissioni consecutive, o su entrambi gli aspetti. In alcuni casi la trasmissione avviene sfruttando più protocolli contemporaneamente, suddividendo le informazioni da trasmettere tra di essi.

Chaffing and Winnowing[2]

La tecnica Chaffing and Winnowing deriva il nome dal processo di separazione della pula dal grano tramite il setaccio. Chaffing and Winnowing garantisce confidenzialità senza utilizzare la crittografia, durante la trasmissione di dati su un canale potenzialmente insicuro.
Sia il mittente (Alice) che il destinatario (Bob) condividono una chiave segreta, ma che non verrà usata se non per scopi di autenticazione: il messaggio verrà comunque trasmesso in chiaro. Alice dividerà il messaggio in bit e ne invierà uno alla volta in pacchetti che contengono anche un numero seriale necessario per poter ricostruire l'ordine del messaggio. Inoltre, insieme a queste informazioni viene inviato un codice di autenticazione del messaggio (MAC) crittografato tramite la chiave che Alice ha condiviso con Bob. Alice consegna i propri pacchetti a Charlie (un utente fidato), che li invia a Bob lungo il canale insicuro, mescolando ai pacchetti costruiti da Alice (il grano) anche dati casuali (la pula). I pacchetti contenenti dati casuali hanno lo stesso numero seriale dei pacchetti di Alice, ma hanno i bit di contenuto invertiti ed un numero casuale al posto del MAC. Bob al momento della ricezione dovrà "setacciare" i messaggi per scoprire se si tratta di informazioni rilevanti o meno, mentre un qualunque utente malevolo che dovesse intercettare la comunicazione non potrebbe distinguere nel flusso i pacchetti di Alice da quelli di Charlie, ammesso che la chiave non sia nota e che Charlie non lasci intendere quali sono i pacchetti autentici a causa dei ritardi di generazione dei suoi pacchetti.

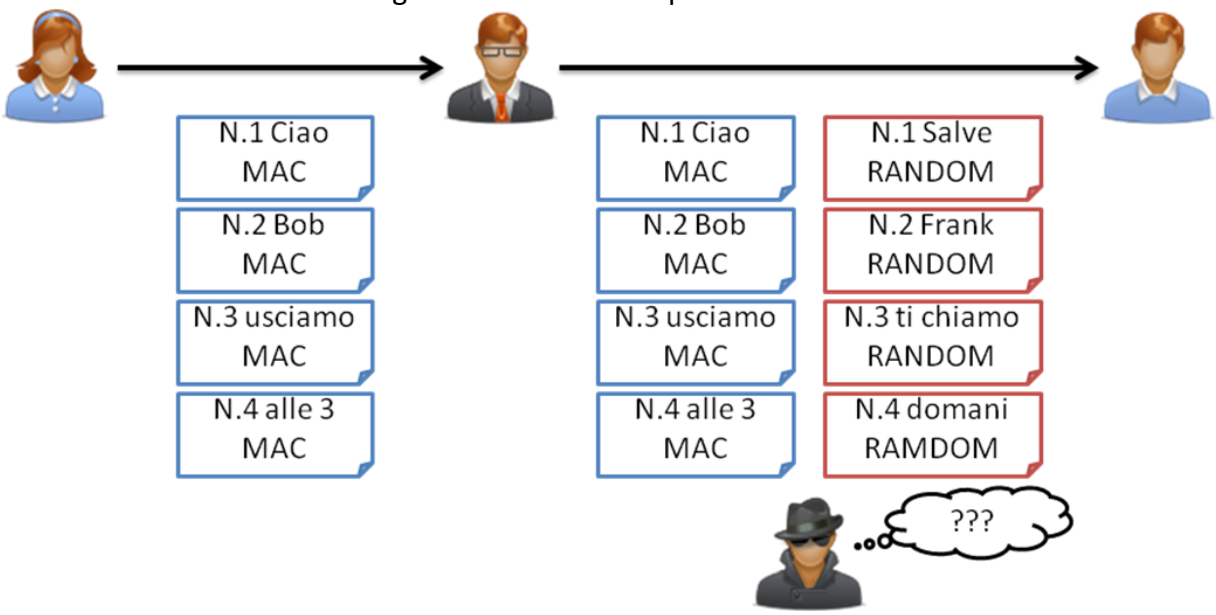


Figura 3 Esempio di comunicazione con Chaffing and Winnowing

Steganografia e watermarking

Il watermark è un marchio impresso in forma esplicita o implicita su un file multimediale o di altro genere allo scopo di identificarlo, di fornire informazioni sulla sua origine o sul suo stato (se è stato modificato), o di tracciarne gli spostamenti. Il watermarking può essere visibile per l'utente finale, o invisibile, ed in questo secondo caso possono essere utilizzate tecniche di steganografia per nascondere il watermark all'interno del documento.
Ad esempio, si è mostrato come è possibile inserire delle informazioni all'interno dei bit meno significativi delle immagini RGB: è evidente come una qualsiasi modifica dell'immagine (ritaglio, ridimensionamento, conversione in altro formato…) possa provocare la perdita o la corruzione delle informazioni inserite: questo semplice sistema potrebbe essere utilizzato per verificare che un'immagine non sia stata modificata.

Canary Trap[3]

La trappola del canarino è un sistema per individuare i responsabili di una fuga di notizie o di informazioni da un ambiente protetto. Se solo un ristretto numero di persone può avere accesso ad un determinato documento, ma il documento diviene pubblico, la steganografia può essere utilizzata per individuare il responsabile della fuga di informazioni: è sufficiente creare tante copie dei dati che devono restare privati quante sono le persone che possono avere accesso a tali informazioni e nascondere (mediante steganografia) una "firma" diversa all'interno di ciascuna copia dei documenti. In questo modo se un documento dovesse essere indebitamente pubblicato, sarà possibile in base alla firma nascosta al suo interno, risalire al responsabile. Una tecnica simile detta Coded Anti Piracy viene utilizzata per marcare in modo discreto le pellicole di film proiettati nei cinema, così da poter risalire alle sale dove le proiezioni vengono riprese illegalmente per riprodurre il film in modo non autorizzato.

Printer steganography[4]

Le comuni stampanti laser a colori prodotte da numerose case introducono un watermark in tutti i documenti stampati: si tratta di punti gialli di dimensioni minuscole, quasi impossibili da individuare ad occhio nudo, disposti in modo da codificare informazioni (tra cui la data e l'ora della stampa ed il numero di serie del dispositivo) utili per poter risalire alla stampante che ha prodotto il documento.

Costellazione di EURione[5]

La costellazione di EURione è un altro esempio di watermarking steganografico: si tratta di un insieme di cinque punti disposti in modo simile alla costellazione di Orione e presente sulle di banconote di diversi paesi attualmente in circolazione. Questo particolare insieme di segni alla vista viene facilmente confuso con delle semplici decorazioni di sfondo ed in alcuni casi è anche mascherato all'interno di altri disegni, ma ha in realtà uno scopo tutt'altro che decorativo ed è infatti utilizzato per impedire la duplicazione delle banconote stesse. Molti software di editing grafico, stampanti, scanner e fotocopiatrici sono infatti in grado di riconoscere le costellazioni di EURione e si rifiutano di procedere con il lavoro, identificando la presenza di una banconota.

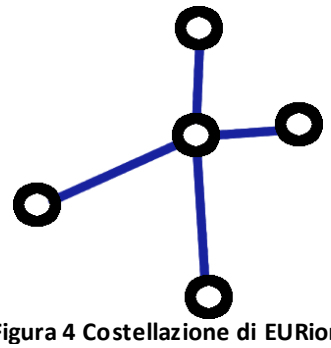


Figura 4 Costellazione di EURione

Steganalisi[6]

La steganalisi è la ricerca dell'esistenza di un messaggio nascosto. Lo scopo principale di questa operazione non è dunque rivelare il contenuto del messaggio, ma solo rilevarne l'esistenza: in alcuni casi questa informazione può essere sufficiente (dopo una più attenta analisi del contenitore) per rivelare anche il contenuto del messaggio. La steganalisi non è affatto un'operazione semplice: un messaggio crittografato è immediatamente riconoscibile (anche se non si conosce l'algoritmo utilizzato per la sua creazione), mentre un documento che contiene all'interno delle informazioni nascoste, appare (o dovrebbe apparire) perfettamente normale. I metodi principali per individuare un contenitore alterato da operazioni di steganografia possono essere:

- Analisi visuale/uditiva,
- Confronto delle proprietà dei file con gli originali,
- Ricerca di tracce nel sistema dell'utente,
- Analisi statistica,
- Ricerca di firme.

Analisi visuale/uditiva

In alcuni casi è possibile ridurre un file contenitore alla sola parte suscettibile a tecniche di steganografia. E' stato mostrato come sia possibile modificare il bit meno significativo nelle immagini per nascondere un messaggio in un contenitore, allo stesso modo durante una ricerca visuale, è possibile eliminare tutti i bit significativi delle immagini ad eccezione dell'ultimo: in questo modo visualizzando l'immagine è possibile individuare ad occhio nudo la presenza di anomalie che possono essere ricondotte a tecniche steganografiche.

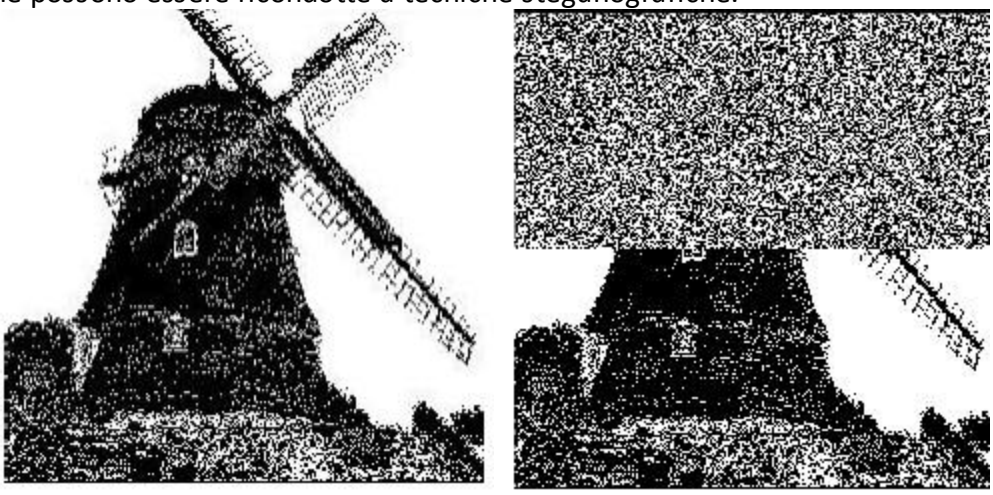


Figura 5 Analisi visuale di un contenitore[7]

Occorre inoltre prestare particolare attenzione alle dimensioni dei documenti da nascondere rispetto a quelle del contenitore: se gli strumenti di steganografia non verificano strettamente che i rapporti tra le due dimensioni rispettino certe regole, è possibile che il contenitore venga modificato fino al punto in cui diventa evidente l'alterazione, anche ad occhio nudo. In altri casi può essere la disattenzione dell'utente a far scegliere un contenitore inadatto, come ad esempio formati che possono essere facilmente corrotti dall'aggiunta di contenuto estraneo.

Confronto delle proprietà dei file (con gli originali)

Se l'utente che ha nascosto il messaggio non è stato particolarmente attento, potrebbe aver scelto contenitori di pubblico dominio (ad esempio disponibili sul web) per nascondere dei dati: questo andrebbe sempre evitato perché una volta noto il contenitore originale, è possibile verificare immediatamente la presenza di contenuto nascosto confrontandone gli attributi (come le dimensioni o il checksum) con l'originale. In altri casi i contenitori potrebbero avere un comportamento noto o atteso in presenza di alcuni dati (ad esempio il formato JPG soffre di particolari distorsioni in caso di netti cambi di contrasto) che può essere alterato dall'inserimento di dati estranei. La modifica di queste caratteristiche attese può essere un indice della presenza di un messaggio nascosto.

Ricerca di tracce nel sistema dell'utente

In alcune situazioni si potrebbe avere a disposizione l'accesso al sistema dell'utente sospettato di aver utilizzato tecniche steganografiche per nascondere delle informazioni (ad esempio nel caso di indagini forensi): se l'utente è stato disattento potrebbe aver conservato copie dei contenitori non modificati che potrebbero essere sfruttati per dei confronti (ad esempio tramite funzioni di hashing) oppure potrebbe aver dimenticato nel suo sistema il programma utilizzato per l'operazione, o ancora potrebbe essere stato il programma stesso ad aver lasciato tracce della sua esecuzione (file di log, tracce nel registro di sistema, file temporanei).

Ricerca di firme

Le applicazioni utilizzate per la steganografia a volte soffrono di problemi strutturali che possono portare all'individuazione dei messaggi nascosti grazie ad esse: il problema più noto sono le cosiddette "firme", ovvero modifiche sistematiche ai file contenitore che vengono eseguite indipendentemente dal tipo di contenitore, dal contenuto e dalla lunghezza degli stessi. Di seguito viene analizzato questo problema all'interno del software Camouflage.

Analisi statistica

(Trattata con maggiore dettaglio nel seguito)

Infrangere la sicurezza di Camouflage[8]

Il software freeware di steganografia Camouflage è divenuto piuttosto popolare intorno l'anno 2000 in ambiente Windows, in quanto permetteva di nascondere e recuperare file in maniera molto semplice direttamente dal menu contestuale, permettendo anche di crittografare i file tramite una password. Il software, sebbene il sito originale sia irraggiungibile, è tuttora reperibile sul web tramite alcuni siti mirror. Non appena il software ottenne una certa notorietà (venne recensito da riviste di informatica divulgative ed in programmi televisivi), diversi hacker si misero a studiarne il comportamento per poter verificare l'effettiva efficacia del suo sistema steganografico. Il fatto che Camouflage possa utilizzare qualunque tipo di file come contenitore desta subito qualche sospetto: solitamente i software steganografici si concentrano su pochi tipi di contenitore e sviluppano tecniche apposite per essi (come si è visto per quanto riguarda le immagini), ma è evidente che non è possibile sviluppare un programma che abbia un diverso algoritmo per ogni possibile tipo di file. Inoltre non ci sono neanche particolari avvertenze per quanto riguarda le dimensioni del contenitore, ma anzi è possibile inserire anche più file di grandi dimensioni

all'interno di un unico contenitore, fatto che non contribuisce a ritenere affidabile la sicurezza di Camouflage.
Il primo passo per tentare di individuare le debolezze di un software per la steganografia consiste nell'esaminare le differenze tra un file contenitore originale e quello modificato dall'inserimento di un contenuto nascosto, inoltre può essere utile anche confrontare diversi file modificati tra loro. Il mezzo per eseguire quest'analisi è un editor esadecimale, in grado di mostrare il contenuto del file in binario. Gli scopi di questa ricerca sono diversi:

- Cercare di capire come opera l'algoritmo, quali parti del contenitore modifica ed in che misura, se opera sequenzialmente o se è randomizzato eccetera;
- Individuare eventuali firme del software.

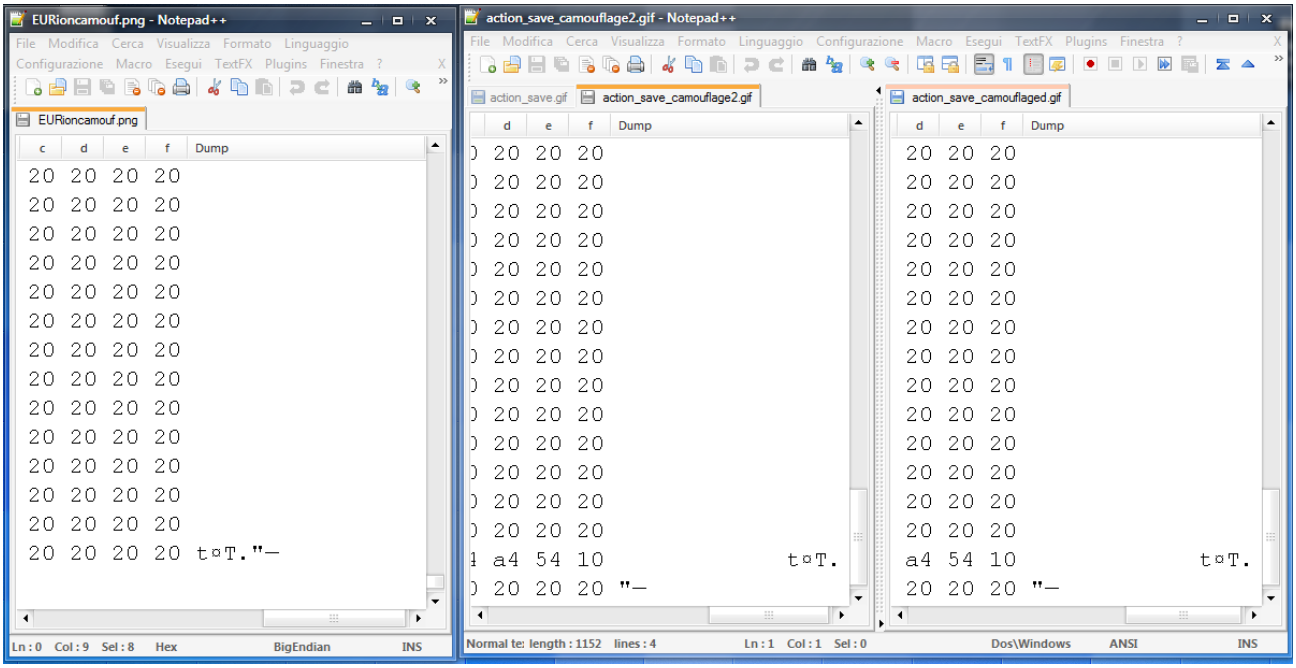


Figura 6 La firma di Camouflage

Come evidenziato dalla figura 6, sono stati messi a confronto tre contenitori dopo essere stati modificati da Camouflage e tutti mostrano una sequenza di simboli subito prima della fine del file (corrispondenti ai valori esadecimali **74 a4 54 10 22 97**): questa è la "firma" del software Camouflage. Una firma è un pericoloso problema per un software di steganografia, perché è facilmente (e velocemente) rintracciabile all'interno di file e non solo indica la presenza di contenuto nascosto, ma rivela anche quale programma è stato utilizzato per l'operazione, e questo potrebbe semplificare enormemente, come vedremo, l'operazione di recupero del contenuto.
Per poter nascondere un file all'interno di un altro è verosimile che da qualche parte all'interno del contenitore venga anche salvata la dimensione del file nascosto, ma visto che Camouflage è in grado di nascondere più di un file e di mantenere anche memoria del nome dei file nascosti, anche questa informazione deve essere contenuta all'interno del contenitore. Eseguendo una ricerca, tramite l'editor esadecimale, della dimensione in base 16 del file che contiene il messaggio, questa può essere rintracciata in due diversi punti, mentre non c'è traccia apparente del nome del file, probabilmente perché è stato crittografato (è interessante notare come questa informazione venga crittografata anche se l'utente non specifica nessuna password).
L'analisi prosegue ora mettendo a confronto un contenitore (di tipo immagine GIF indicizzata) con la versione modificata da Camouflage.

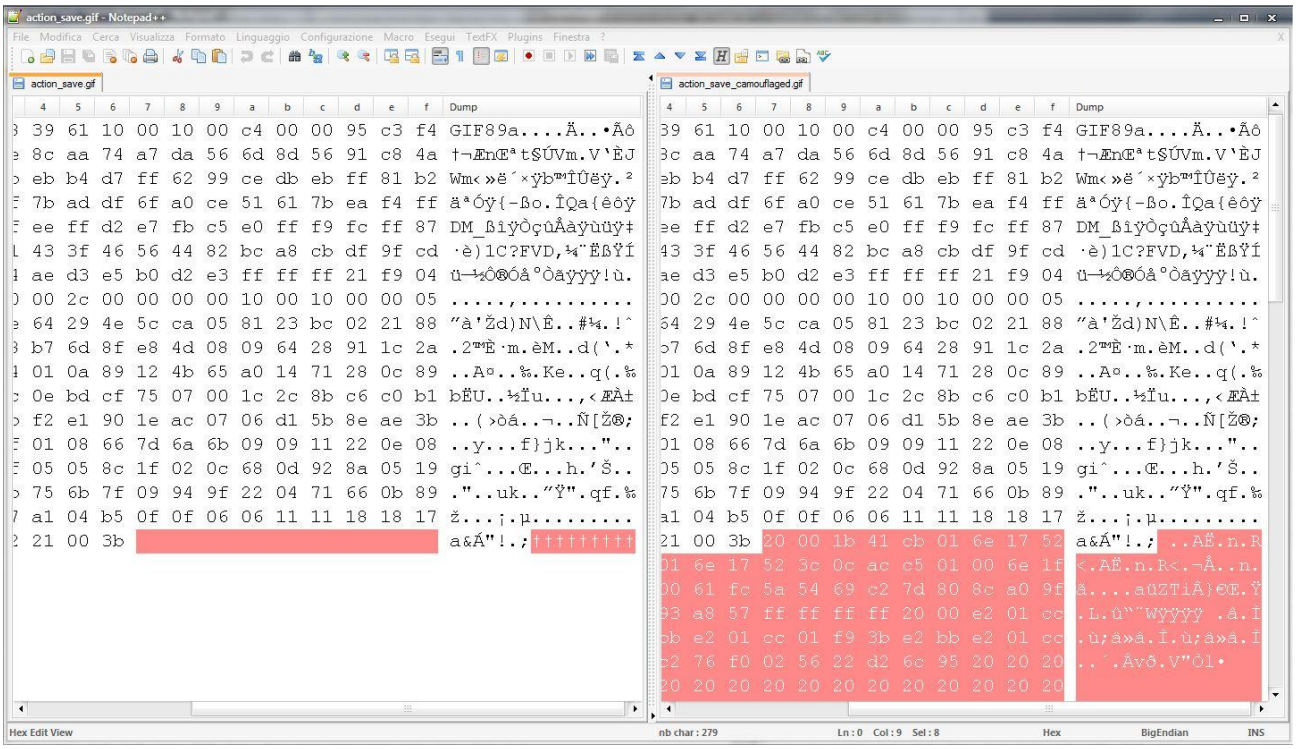


Figura 7 Una immagine prima (a sinistra) e dopo (a destra) l'inserimento di un messaggio.

Evidenziando le differenze tra i file si vede come niente dell'immagine sia stato in realtà modificato: il nuovo file è esattamente uguale al contenitore originale per quanto riguarda la prima parte, ma gli sono stati aggiunti dei dati in coda. Questo dimostra che Camouflage utilizza una algoritmo "standard" per qualunque tipo di contenitore, nonostante si sia visto come per le immagini indicizzate esistano metodi particolari per inserire un messaggio nella palette. Per verificare che non sia stato un caso sono state effettuate altre prove, di cui quella che evidenzia in modo migliore la debolezza di questo approccio è mostrata nella figura 8, tentando di nascondere il nostro messaggio all'interno di file di testo: neanche in questo caso è stata sfruttata nessuna tecnica particolare studiata appositamente per il formato del contenitore ed aprendo il documento con un normale editor testuale appare evidente il flusso inserito da Camouflage. Una possibile tecnica, utilizzata da altri tool steganografici[9], per contenitori di tipo testuale consiste nell'aggiungere spazi e righe vuote in fondo al file, visto che questi solitamente non vengono mostrate dagli editor e quindi possono sfuggire facilmente ad un'analisi superficiale.

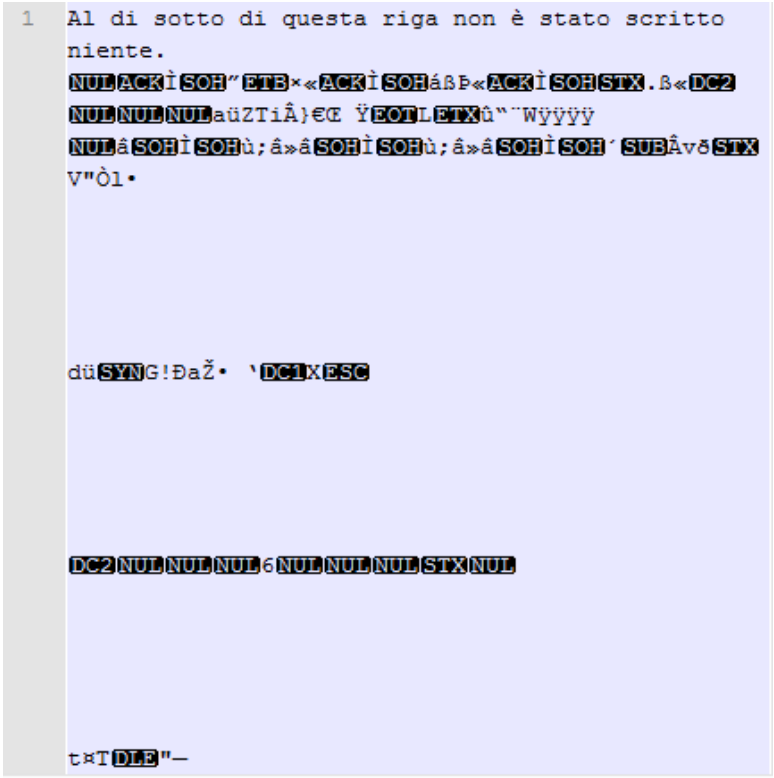


Figura 8 Un file testuale dopo l'inserimento di un messaggio nascosto

Si noti come la firma di Camouflage è chiaramente visibile in fondo al file modificato. La tecnica di Camouflage, che consiste nell'accodare un flusso di dati nel contenitore funziona nella maggior parte dei casi perché quasi tutti i formati di file hanno una particolare struttura al loro interno che delimita l'inizio e la fine di ogni sezione informativa: i dati inseriti in coda finiranno verosimilmente dopo un delimitatore di "fine del file" e verranno con buona probabilità ignorati dai programmi di visualizzazione e modifica del contenitore.
Nonostante la steganalisi finora abbia dato esiti positivi (possiamo individuare la presenza di contenuto steganografato mediante Camouflage all'interno di qualunque file, cercando la firma del software), se è stata inserita una password al momento dell'inserimento del messaggio nel file, non siamo comunque in grado di ottenere il testo in chiaro. A questo punto un ulteriore confronto di due contenitori modificati potrebbe fornire qualche indizio utile su come operare per infrangere la sicurezza del software: questa volta utilizziamo due copie dello stesso contenitore per nascondere lo stesso messaggio in un caso senza fornire una password, e nell'altro inserendone una, quindi confrontiamo gli output del software. Sorprendentemente le differenze tra i file sono minime, il blocco di dati aggiunto da Camouflage in coda al contenitore è quasi identico tra i due file, ad eccezione di pochi byte, il che lascia ragionevolmente supporre che il messaggio non venga effettivamente crittografato utilizzando la password che ci viene richiesta.

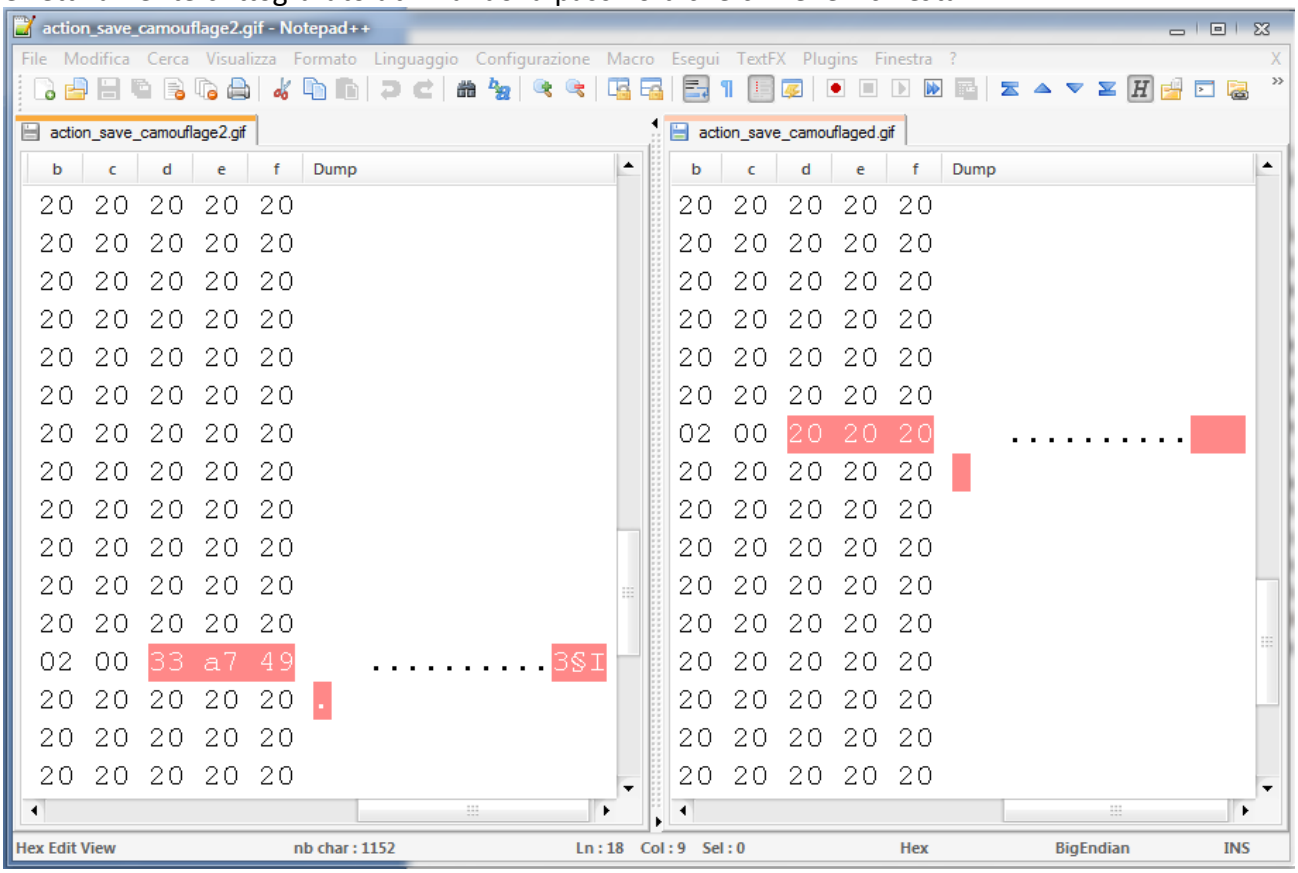


Figura 9 Contenitore con password (a sinistra) e senza (a destra)

Variando la lunghezza della password infatti vediamo che le differenze tra i due file si modificano in modo direttamente dipendente dalla lunghezza e dal contenuto della password stessa, il che porta a supporre che la password, anziché servire per crittografare il messaggio, venga scritta nel contenitore e poi confrontata con quella immessa dall'utente al momento dell'estrazione dei documenti nascosti: questo significa anche che analizzando il codice assembly dell'eseguibile è possibile risalire all'algoritmo utilizzato per crittografare i contenuti (pratica nota come reverse

engineering) ed invertire il processo per ottenere i documenti originali. In realtà non è neanche necessario effettuare questa operazione (piuttosto complicata), in quanto analizzando ulteriormente la struttura del file è possibile stabilire in quale posizione del blocco di dati aggiunto da Camouflage la password è stata salvata e, sfruttando solamente l'editor esadecimale, rimuoverla. Questo spiega perché, dopo l'iniziale attenzione rivolta a questo software freeware ci si sia spostati altrove per cercare strumenti steganografici efficaci.

Attacchi basati su analisi statistiche delle immagini

La steganalisi basata sulle analisi statistiche non solo offre dei sofisticati metodi per contrastare con successo, in parecchi casi, la steganografia delle immagini, ma viene praticata per identificare e valutare le debolezze di un sistema steganografico al fine di migliorarne la sicurezza.

A questo proposito è opportuno soffermarsi sul significato di sicurezza di un sistema steganografico.
In crittografia è il principio di Kerckhoff a definire un sistema crittografico sicuro: se la chiave è confidenziale può essere reso pubblico l'algoritmo di cifratura.
Dunque crittografando il messaggio prima di steganografarlo e rendendo pubblico l'algoritmo steganografico, si potrebbe garantire la confidenzialità; ma in tal maniera verrebbe meno l'obiettivo principale della steganografia, che è quello di non essere rilevabile in se per se.
E' evidente quindi che la definizione di sicurezza non può avere lo stesso significato di quella che si dà per un crittosistema. In letteratura sono state proposte numerose definizioni di sicurezza steganografica. A grandi linee si può affermare che un sistema steganografico risulta sicuro se le immagini steganografate conservano le stesse proprietà statistiche di quelli originali. In generale si assume che l'attaccante abbia illimitate capacità computazionali e dettagliate conoscenze statistiche delle immagini contenitore.
Tuttavia mentre è abbastanza ragionevole assumere che tutti gli aspetti del meccanismo crittografico siano noti ad un attaccante, è meno plausibile assumere che egli conosca in dettaglio le proprietà delle immagini vettore. Infatti, essendo infinite le immagini che possono fungere da mezzo, non è possibile formalizzare un modello statistico che descriva con precisione tutte le caratteristiche di un'immagine vettore tipo.
In realtà la scelta dell'immagine (formato e contenuto) costituisce un fattore determinante per la sicurezza. Ad esempio la compressione del formato JPEG risulta estremamente robusta a piccole modifiche come l'inserimento di un messaggio; ciò significa che è possibile risalire all'immagine originale a partire da una che si ritiene sospetta ed operare un semplice confronto. Vi sono inoltre alcuni formati che suggeriscono di per se il software e quindi l'algoritmo steganografico usato. Per quanto riguarda il contenuto, da evitare sono le immagini con aree monocolore o troppo regolari, mentre si consigliano quelle complesse da un punto di vista visivo e che presentano rumore intrinseco.
Per tutti questi motivi l'utilità delle attuali definizioni di sicurezza è molto limitata.

LSB Embedding

Prima di andare a vedere in che consistono i principali attacchi alla steganografia delle immagini, è necessario capire il funzionamento dei metodi steganografici più diffusi.
Innanzi tutto fissiamo delle notazioni che saranno utili nel seguito:

- Con la sigla LSB (Least Significance Bit) si intende l'ultimo bit significativo. Come già detto nel primo paragrafo, le immagini sono vere e proprie matrici di pixel ognuno dei quali può contenere uno oppure tre numeri binari di 8 bit ciascuno.
  Al primo tipo appartengono le immagini in formato GIF, BMP scala di grigi 8-bit e JPEG; in tali casi il valore del pixel rappresenta rispettivamente un indice della palette, una gradazione di grigio, o un coefficiente DCT quantizzato frutto della compressione con la Trasformata Discreta del Coseno.

Al secondo tipo appartengono le immagini codificate in RGB e i tre valori contenuti nel pixel indicano rispettivamente una gradazione di rosso, di verde e di blu.

- Il Bit Plane di un'immagine è l'insieme ordinato dei bit aventi la stessa posizione (es. LSB) nei rispettivi valori dei pixel che compongono l'immagine.

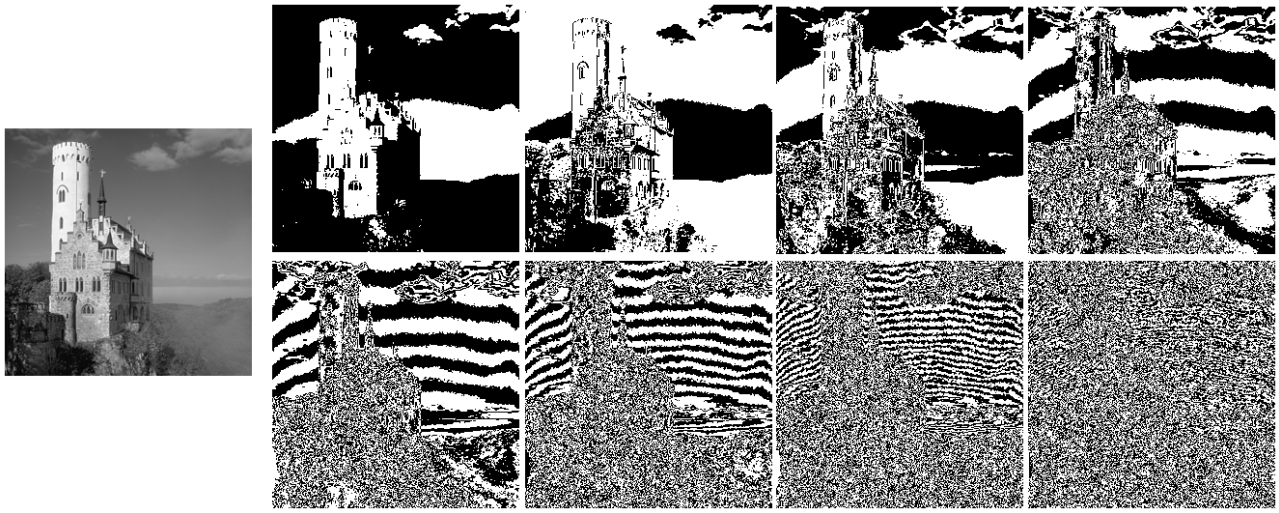


Figura 10 Bit Planes dal più al meno significativo.

Le tecniche steganografiche più diffuse si basano sul cosiddetto **LSB embedding**: il messaggio da nascondere viene convertito in una stringa di bit, i quali vengono "incorporati" uno ad uno all'interno dell'immagine vettore, sfruttando l'ultima cifra significativa del valore contenuto nei pixel.
Si possono distinguere due tipi di LSB embedding:

- **LSB replacement**: i bit del messaggio *rimpiazzano* il LSB del valore dei pixel d'appoggio. In tal caso viene modificato solo il LSB plane.
- **LSB maching**: si fa *corrispondere* il LSB del valore del pixel al bit del messaggio da inserire, incrementando o decrementando di uno (secondo specifici criteri che si differenziano tra i vari stego-sistemi di questa classe) il valore del pixel di supporto. In tal caso vengono modificati anche gli altri bit plane. Questa tipologia è più difficile da individuare.

I bit del messaggio possono essere inseriti nell'immagine in maniera:

- **Sequenziale**: partendo dalla sinistra in alto e procedendo per righe, viene utilizzato ogni pixel fino a quando il messaggio non viene esaurito.
- **Selettiva**: evitando i pixel che contengono valori particolari: ad esempio l'algoritmo Jsteg evita di modificare i coefficienti pari a 1 o a 0 (in binario 00000001 e 00000000).
- **Pseudo-random**: utilizzando un generatore di numeri pseudo-casuali, i pixel di supporto vengono selezionati lungo un percorso casuale (che dipenderà da una chiave fornita dall'utente) all'interno dell'immagine. Questa tecnica è più sofisticata.

Attacchi statistici

Ciò che accomuna tutti gli attacchi che rientrano nella tipologia che si sta trattando, è la necessità di individuare, anche in funzione del formato, una proprietà dell'immagine che statisticamente viene compromessa o alterata da uno o più processi steganografici.

Una volta scelta la proprietà da analizzare (può trattarsi dello spettro cromatico, di una misura del rumore o dell'entropia, della simmetria degli istogrammi di colore ecc) si fissano le cosiddette zero-message hyphothesis; in altre parole si tenta di formalizzare una o più caratteristiche che, statisticamente, sembra avere un'immagine vettore tipo, priva di contenuti nascosti in relazione alla proprietà scelta (questo passaggio risulta delicato per i motivi già sopra detti). Ovviamente ciò risulta superfluo nel caso in cui l'attaccante sia in grado di risalire in qualche maniera all'originale di un'immagine sospetta; in tale caso, infatti, è sufficiente un semplice confronto.
A questo punto si estrapolano dall'immagine steganografata i dati necessari a stimare la proprietà in questione.
Infine l'uso di appositi strumenti statistico matematici permette da un lato di quantificare in che misura le proprietà rilevate si discostano da quelle "previste" (zero-message hypothesis), dall'altro di dare una stima della lunghezza del messaggio inserito.
Infatti, l'alterazione di queste proprietà non solo costituisce di per se un indicatore di presenza di contenuti nascosti, ma spesso dipende funzionalmente dalla lunghezza del messaggio.

Attacco χ^2 (chi-quadro)

Questo attacco si applica a immagini che hanno subito un processo di LSB replacement sequenziale e con delle piccole variazioni funziona anche nel caso di LSB replacement sequenziale selettivo (Jsteg).
Vediamo che cosa succede nel caso particolare dell'algoritmo EzStego.
Ezstego è uno stego-sistema che applica il metodo LSB replacement sequenziale alle immagini in formato GIF. Prima di iniziare con la modifica dell'immagine, EzStego crea una copia della palette ordinandola (reindicizzandola) in maniera tale che sia difficile stabilire ad occhio nudo la differenza tra due colori adiacenti. In altre parole colori aventi indici successivi debbono essere visivamente simili. Se interpretiamo ogni colore come un punto in uno spazio tri-dimensionale (cubo dei colori RGB), la figura 11 è un esempio di come avviene questo ordinamento.

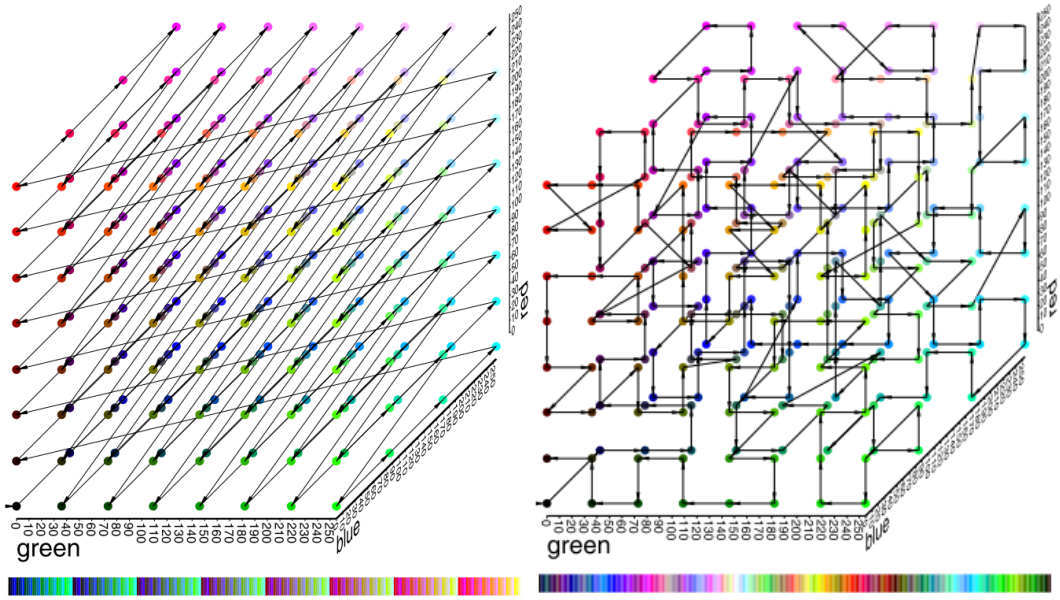


Figura 11 Sequenza dei colori attraverso il cubo RGB prima e dopo l'ordinamento di EzStego.

Lo scopo di tale riordinamento è di rendere l'immagine immune da alterazioni percepibili ad occhio nudo dovute alla modifica del LSB.

L'attacco χ^2 è basato sul fatto che durante il processo di LSB replacement si vengono a formare delle coppie di valori (PoVs: Pairs of Values) che si interscambiano.
D'ora in poi per semplicità si farà riferimento alla Figura 12.

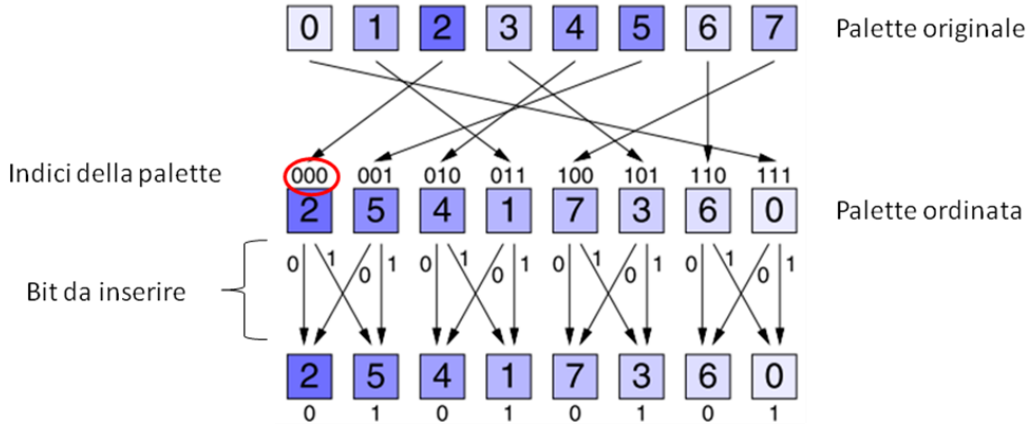


Figura 12

Se ad esempio durante il processo di embedding si incontra un pixel contenente indice di colore 110 e si vuole inserire il bit 0, il valore del pixel resterà invariato, mentre se si vuole inserire il bit 1 l'ultima cifra significativa verrà modificata e il valore del pixel diverrà 111. Viceversa se un pixel contiene l'indice 111 e l'ultimo bit viene modificato il valore diverrà 110. Dunque si vengono a creare delle coppie di indici: pari e dispari successivo.

$$0 \leftrightarrow 1 \ , \ 2 \leftrightarrow 3 \ , \ 4 \leftrightarrow 5 \ , \ \dots \ , \ 2i \leftrightarrow 2i+1 \ , \ \dots$$

Si è rilevato statisticamente che per un'immagine non originale, i due valori di ogni coppia si presentano con frequenze differenti. Invece dopo il processo di embedding, si osserva che queste frequenze tendono a pareggiarsi.

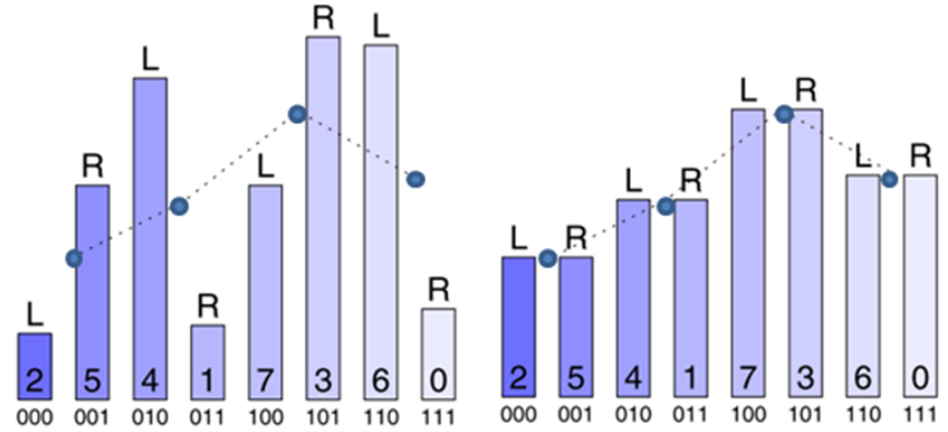


Figura 13 Istogrammi delle frequenze dei colori prima e dopo l'immissione di un messaggio con EzStego.

Ciò che invece resta immutata è la somma di queste due frequenze.
Convinciamocene con il seguente esempio. Consideriamo la PoV (100,101). Se durante il processo di embedding si incontra un pixel contenente indice di colore 100 e l'ultima cifra significativa viene modificata ottenendo 101, allora la frequenza del colore di indice 100 diminuirà di uno mentre quella del colore di indice 101 aumenterà di uno. Viceversa se l'ultimo bit di 101 viene modificato ottenendo 100, la frequenza del primo diminuirà di uno mentre quella del secondo aumenterà di

uno. Dunque per ogni coppia la somma delle frequenze non viene intaccata dal processo di immissione.

Se poniamo x_i il numero di occorrenze del colore di indice 2i , y_i il numero di occorrenze del colore di indice 2i+1 e

$$z_i = \frac{x_i - y_i}{2}$$

Allora z_i è il numero di occorrenze teoricamente previste per un'immagine steganografata.

In generale, per valutare quantitativamente la bontà dell'adattamento delle frequenze osservate alle frequenze attese si utilizza la *statistica test chi-quadro*. Allora supponendo che vi siano N Povs con frequenza $x_k > 4$ (condizione di frequenza minima) si calcola

$$\chi^2_{N-1} = \sum_{k=1}^{N-1} \frac{(x_k - z_k)^2}{z_k}$$

E' detta *statistica test chi-quadro con (N-1) gradi di libertà*.
Osserviamo che nella formula, all'interno della sommatoria, vi è una sottrazione. Quindi intuitivamente ci aspettiamo che se x_k e z_k sono circa uguali (l'immagine ha le caratteristiche di una stego-immagine) allora il valore di χ^2_{N-1} sarà piccolo, mentre se x_k e z_k sono molto diversi (l'immagine è naturale) allora il valore di χ^2_{N-1} sarà grande.
L'ultimo passo è quello di misurare la probabilità che un qualche messaggio sia stato incorporato. Per fare questo si calcola il cosiddetto p-value del test, che si ottiene integrando la funzione densità della distribuzione χ^2_{N-1} e misura la significanza statistica del test.

$$p = 1 - \frac{1}{2^{\frac{N-1}{2}}\Gamma(\frac{N-1}{2})} \int_0^{\chi^2_{N-1}} e^{-\frac{x}{2}} x^{\frac{N-1}{2}-1} dx$$

Il valore di p fornisce la probabilità che una data immagine sia stata steganografata (probabilità di embedding). Disegnando un grafico che esprime questa probabilità al variare della percentuale di immagine scansionata, si osserva in corrispondenza della fine del messaggio nascosto, un brusco cambiamento nell'andamento della probabilità. Si può dare quindi una stima della lunghezza del messaggio.

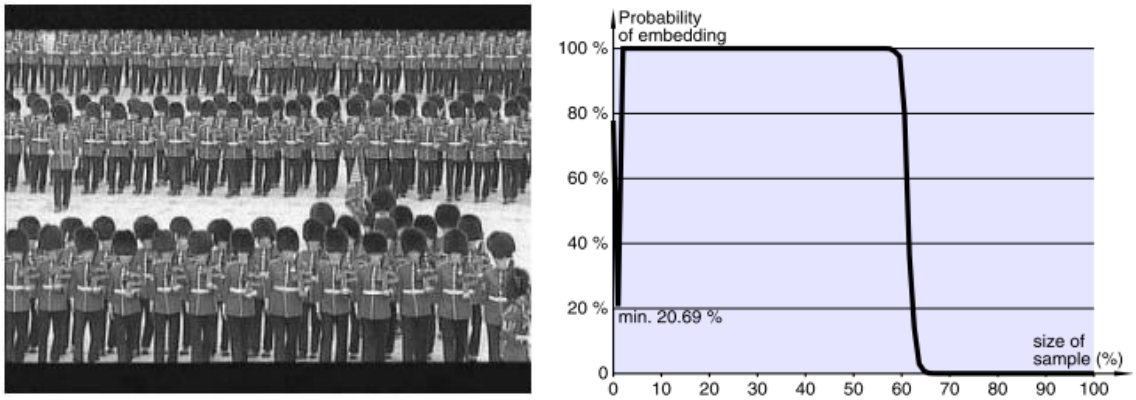


Figura 14

Attacco Chi-Quadro Generalizzato

Questo attacco è applicabile ad immagini steganografate tramite LSB replacement pseudo-random. Invece di scansionare l'immagine in maniera sequenziale, a partire da una posizione fissa, aumentando progressivamente la percentuale esaminata, in questo caso vengono selezionate delle vere e proprie "finestre" di dimensioni opportunamente fissate, e fatte scorrere lungo l'immagine. La posizione cambia dell'1% ad ogni misurazione. Di volta in volta si registra il valore di p. Alla fine si esegue una somma con riscalamento di tutte le probabilità registrate.

Metodo RS o delle doppie statistiche (Dual statistics)

E' applicabile ad immagini steganografate tramite LSB replacement Pseudo-Random.
Mentre i metodi precedenti concentrano l'attenzione su il LSB-plane, questa analisi cerca di dare un certo peso anche alla posizione in cui si trova un determinato pixel all'interno dell'immagine. L'idea è che il LSB plane è connesso in qualche modo agli altri sette bit plane.

Figura 15

Questa debole relazione in un certo senso misura la capacità di una determinata immagine di non perdere dati e può essere catturata matematicamente tramite la combinazione di due accurate statistiche che ci permettono di quantificare come viene influenzata questa connessione dal processo steganografico.

Analisi degli istogrammi

Gli attacchi che rientrano in questa categoria sono numerosi e differenti tra loro. I più sofisticati sono applicabili ad immagini steganografate con LSB maching Pseudo-random.
Viene eseguita una stima dell'immagine originale e comparata con quella steganografata.
Nel caso del formato JPEG è possibile risalire, con buona approssimazione, all'immagine originale tramite un processo detto *calibrazione* (decompressione e ricompressione con una struttura a blocchi differente) e mettere a confronto l'immagine ottenuta con quella da analizzare per risalire alla lunghezza del messaggio inserito. Ad esempio alcuni attacchi sono basati sullo studio della perdita di simmetria degli istogrammi che spesso segue ad un processo steganografico.

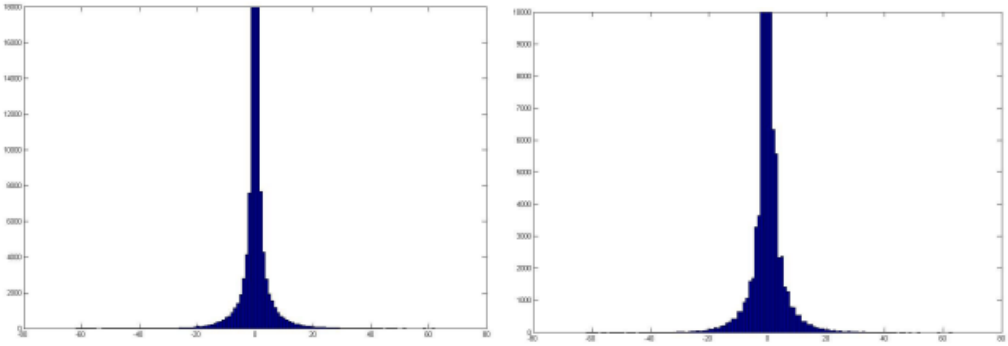


Figura 16 Istogrammi dei coefficienti AC di un'immagine Jpeg prima e dopo il processo di embedding tramite Jsteg.

Blind analysis

A differenza di tutti i metodi precedentemente esposti la Blind Analysis non è specifica per un determinato algoritmo steganografico ma è pensata per funzionare per tutte le tecniche di embedding e formati di immagine.
Alla base di questa disciplina, c'è la ricerca di un set di proprietà, grandezze *sensibili* al processo steganografico (cioè proprietà che statisticamente non vengono preservate). Si studia il comportamento di queste grandezze su centinaia d'immagini campione, provenienti da database ufficiali, al variare della lunghezza del messaggio da nascondere e del processo steganografico. Tramite l'utilizzo di statistiche del primo-ordine (analisi degli istogrammi) o di ordine superiore, vengono sviluppati dei veri e propri rilevatori in grado di distinguere tra immagini naturali o meno e di riconoscere l'algoritmo di embedding utilizzato.

Metodi Steganografici migliorati

Di pari passo con l'evoluzione della steganalisi va quella della stenografia.
I metodi steganografici più recenti tentano di ostacolare o rendere impossibile allo steganalista di eseguire una stima accurata dell'immagine originale. Ciò viene fatto randomizzando la scelta dei pixel, della posizione all'interno dell'immagine vettore e addirittura del metodo steganografico. Infine alcuni metodi sono in grado incorporare un messaggio simulando il rumore intrinseco.

Le icone utilizzate sono distribuite sotto licenza Creative Commons da:

- famfamfam.com (http://creativecommons.org/licenses/by/2.5/)
- pixel-mixer.com (http://creativecommons.org/licenses/by-sa/3.0/)

Riferimenti

[1] Chvarkova et al. *Steganography: Digital Data Embedding Techniques*. (http://scientist.by/index.php?option=com_content&view=article&id=37%3Asteganography-digital-data-embedding-techniques&catid=9&Itemid=27&limitstart=5 online il: 24/04/11).

[2]Ronald L. Rivest, Chaffing and Winnowing: Confidentiality without Encryption (http://people.csail.mit.edu/rivest/chaffing-980701.txt last access: 24/04/11).

[3] Canary Trap, Wikipedia the free encyclopedia (http://en.wikipedia.org/wiki/Canary_trap online il: 24/04/11)

[4] Electronic Frontier Foundation, List of Printers Which Do or Do Not Display Tracking Dots (https://www.eff.org/pages/list-printers-which-do-or-do-not-display-tracking-dots online il: 24/04/11).

DocuColor Tracking Dot Encoding Guide (http://www.eff.org/Privacy/printers/docucolor/ online il: 24/04/11).

[5] Markus Kuhn, The EURion constellation (http://www.cl.cam.ac.uk/~mgk25/eurion.pdf online il: 24/04/2011)

[6] Steganography, Steganalysis, & Cryptanalysis, Michael T. Raggo (https://www.defcon.org/html/links/dc-archives/dc-12-archive.html online il: 28/04/2011)

[7] Immagine da: Westfeld, A. and Pfitzmann, A. (2000) 'Attacks on Steganographic Systems', 3rd International Workshop. Lecture Notes in Computer Science, Vol.1768. Springer-Verlag, Berlin Heidelberg New York

[8] Breaking a steganography software: Camouflage, by Guillermito (http://www.guillermito2.net/stegano/camouflage/index.html online il 29/04/2011)

[9] SNOW, Steganographic Nature Of Whitespaces (http://www.darkside.com.au/snow/index.html online il 29/04/2011)

[10] Westfeld, A., Pfitzmann, A.: Attacks on steganographic systems. In: Proc. Information Hiding Workshop. Volume 1768 of Springer LNCS. (1999) 61–76

[11] Fridrich, J., Goljan, M.: Practical steganalysis of digital images – state of the art. In Delp III, E.J., Wong, P.W., eds.: Security and Watermarking of Multimedia Contents IV. Volume 4675 of Proc. SPIE. (2002) 1–13

[12] Fridrich, J., Goljan, M., Soukal, D.: Higher-order statistical steganalysis of palette images. In Delp III, E.J., Wong, P.W., eds.: Security and Watermarking of Multimedia Contents V. Volume 5020 of Proc. SPIE. (2003) 178–190

[13] Andrew D. Ker: Improved Detection of LSB Steganography in Grayscale Images. Oxford University Computing Laboratory, Parks Road, Oxford OX1 3QD, England

[14] Christy A. Stanley: Pairs of Values and the Chi-squared Attack. Department of Mathematics, Iowa State University

[15] Xu Mankun, Li Tianyun, Ping Xijian: Steganalysis Of LSB Matching Based On Histogram Features in Grayscale Image.Dept. of Information Science, Univ. of Information Engineering

[16] Ergong Zheng, Xijian Ping, Tao Zhang, Gang Xiong: Steganalysis of LSB matching based on local variance histogram. Zhengzhou Information Science and Technology Institute.

[17] Jessica Fridrich, Miroslav Goljan, and Rui Du: Detecting LSB Steganography in Color and Gray-Scale Images. State University of New York, Binghamton