Università degli Studi di Roma
"La Sapienza"

Facoltà di Ingegneria

Corso di Laurea Specialistica in
Ingegneria Informatica

anno accademico 2004-05

Steganografia e Crittografia tramite Stereo Visione

Relatore Prof. Luca Iocchi

Autore Domenico Bloisi

« Le cose non si vedono per come sono,
ma per come è chi le osserva. »

FABIO VOLO

Indice dei contenuti

Ringraziamenti

Le pagine a seguire rappresentano il frutto di due anni di lavoro, racchiudono gran parte di quello che ho imparato frequentando l'università e, soprattutto, mi danno la possibilità di raccontare la mia idea. Questa tesi nasce, infatti, dalla mia intuizione di usare la stereo visione per celare un messaggio in una coppia di immagini, un modo finora inedito di sfruttare tale tecnica. Sentitamente ringrazio il professor Luca Iocchi per avermi dato la grande possibilità di mettere nero su bianco questa intuizione, indicandomi il modo corretto per trasformare quanto avevo in mente in una valida trattazione scientifica.
Ringrazio di cuore mamma Pina, papà Franco, Viviana e Chiara perché mi vogliono bene e me lo dimostrano sempre.
Sono grato a Bernardino, Egidio e Giovanni perché sopportarmi per sei anni non deve essere stato facile.
Dico grazie ad Andrea perché solo lui riesce a metterti quell'ansia che dà un fastidio inenarrabile e fa sembrare tutto complicatissimo.
Con chi avrà la pazienza di leggere questo lavoro, mi scuso fin d'ora per gli errori commessi.

Roma, 20 maggio 2006

Domenico Bloisi

P. S. Judclh Dggr shufkh pl vhl vhpsuh ylflqd

Introduzione

Introduzione

Il problema

La steganografia è un modo poco conosciuto di proteggere la confidenzialità di un qualsiasi insieme di dati; normalmente, è la crittografia ad essere usata per le comunicazioni riservate. Molti algoritmi crittografici – la sicurezza dei quali è stata lungamente e profondamente studiata negli anni – sono oggi ampiamente disponibili. Tuttavia, a differenza della steganografia, le tecniche crittografiche generano messaggi che sono facilmente *riconoscibili* come messaggi cifrati, sebbene il loro contenuto originario rimanga confidenziale (è, infatti, molto difficile e dispendioso decrittare un messaggio cifrato con i moderni algoritmi). Questa "riconoscibilità" dei file cifrati li rende inutilizzabili in contesti in cui sia previsto un controllo selettivo sul traffico informativo (si pensi ad alcuni regimi che vietano l'uso della crittografia).
La steganografia, termine composto da due parole di origine greca, *stèganos* (che significa nascosto) e *gràfein* (che significa scrivere), è l'insieme dei metodi e delle tecniche che permettono di inserire un messaggio confidenziale (che si vuole rimanga segreto) all'interno di un altro messaggio, molto più esteso (ad esempio, una immagine, un file mp3, una pagina web) che funge da *contenitore* e che si vuole risulti di pubblico dominio. L'obiettivo è quello di modificare il contenitore in modo impercettibile, in modo che esso non sollevi alcun sospetto se sottoposto ad analisi – non deve essere possibile né risalire al contenuto del messaggio segreto né, tanto meno, all'esistenza stessa di una comunicazione riservata. Nessuno deve poter sospettare che si voglia trasmettere materiale riservato: la steganografia *fallisce* nel momento in cui la trasmissione viene scoperta, anche se non si è in grado di decrittarne il contenuto. L'esempio di un sistema steganografico che adopera immagini come contenitori è mostrato in figura 1.

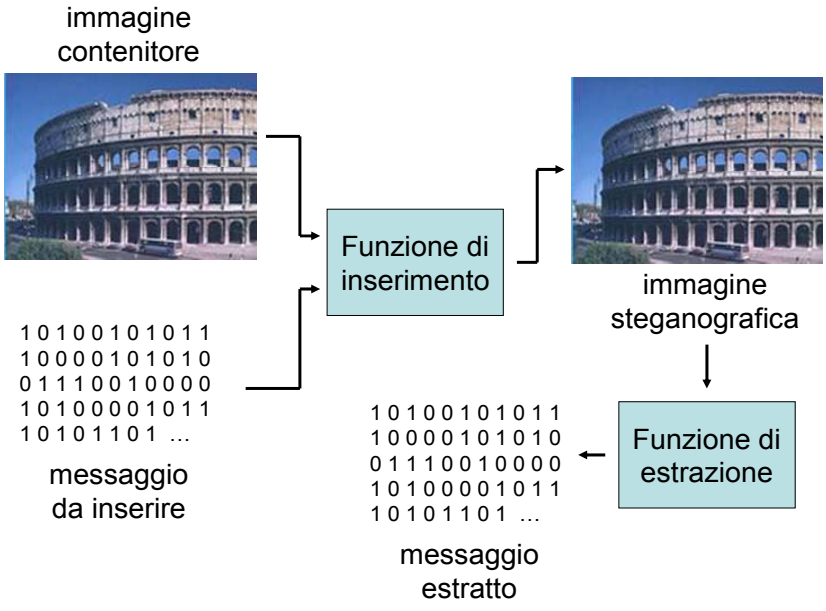


Figura 1 – Un sistema steganografico

Il mittente (*sender*) crea l'immagine steganografica usando la funzione di inserimento la quale presenta due parametri in input:

1. una immagine contenitore (*cover image*) contenente una qualche forma di informazione ridondante o casuale (ad esempio, del rumore),

2. il messaggio da inserire (*messaggio segreto*).

I dati multimediali, come brani audio e video, sono eccellenti contenitori: infatti, a seguito della digitalizzazione, essi contengono del rumore di quantizzazione che fornisce lo spazio di manovra necessario all'inserimento dei dati; inoltre, la compressione con perdita (*lossy*) è in grado di introdurre ulteriori quantità di rumore.
Usando la funzione di estrazione, il destinatario (*receiver*) è in grado di riprodurre il messaggio segreto a partire dall'immagine steganografica (*stego image*).
È molto importante che un file steganografico (*stego file*) conservi le stesse proprietà statistiche che caratterizzano il file contenitore, in modo che sia difficile appurare la presenza di anomalie prodotte dall'inserimento del messaggio segreto. Conseguentemente, un (potenziale) messaggio deve poter essere letto sia dal file steganografico, sia dal file contenitore. Come regola involabile, un messaggio letto dal file steganografico non deve risultare statisticamente differente dal potenziale messaggio letto dal file contenitore, altrimenti il sistema steganografico non sarebbe sicuro.

Gli obiettivi

I sistemi steganografici sviluppati finora non si occupano di effettuare direttamente una *cifratura forte* del messaggio che inseriscono nello stego file. Se si vuol conservare veramente al sicuro il contenuto del messaggio segreto, bisogna prima cifrarlo con algoritmi propriamente crittografici e, solo dopo questa cifratura classica, fornirlo in input alla funzione di inserimento del sistema steganografico.
Questa pre-elaborazione crittografica comporta l'aumento dell'entropia dei dati che devono essere inseriti nel file contenitore, rendendo più facile individuare la presenza di un contenuto occulto all'interno dello stego file. Di fatto, messaggi ad alta entropia presentano, a differenza dei messaggi non cifrati, una distribuzione uniforme dei bit tra uni e zeri, rendendo necessario trasformare il contenitore (che ha bassa entropia, non essendo cifrato) pesantemente. Tutto ciò ostacola il compito della steganografia perchè tale trasformazione va a distruggere alcune note caratteristiche del cover.
Poiché esiste un modo molto semplice (l'analisi statistica [4, 19]) per verificare che una immagine abbia o meno quelle caratteristiche, esiste una strategia di ricerca valida e robusta per discernere tra file sospetti e non, costringendo, in definitiva, al fallimento un sistema steganografico che compia una pre-elaborazione crittografica.
L'obiettivo principale di questa tesi è quello di superare tale limite, in particolare di:

1. definire un innovativo algoritmo che effettui steganografia e crittografia allo stesso tempo. Questa procedura dovrà avere prestazioni comparabili ai più

moderni sistemi steganografici e permettere di cifrare il messaggio con risultati, in termini di sicurezza, paragonabili ai metodi crittografici più sicuri;

2. implementare questo sistema per permetterne la validazione attraverso una serie di test comparativi con le tecniche steganografiche e crittografiche note;

3. mostrare alcuni possibili scenari d'uso per tale sistema.

I risultati

La presente tesi propone un nuovo paradigma di sintesi tra steganografia e crittografia che sfrutta la *stereo visione* come ideale anello di congiunzione tra esse.
La stereo visione è un campo della visione artificiale (*computer vision*) il cui fine concerne l'ottenimento di una mappa di profondità a partire da una coppia di immagini stereo, mediante la ricerca di corrispondenza dei punti coniugati delle due immagini che permette, poi, di effettuare la triangolazione [17].
La stereo visione viene attualmente impiegata per la ricostruzione 3D, la navigazione di robot mobili, la realizzazione di carte geografiche tridimensionali a partire da foto aeree, etc…; non siamo a conoscenza di esempi di impiego della stereo visione in combinazione con crittografia e steganografia.
Le pagine di questo testo mostrano come l'idea base della stereo visione (il match di due immagini stereo) possa essere applicata con successo nell'implementazione di un sistema steganografico per mezzo del nuovo concetto di *key image* (figura 2).

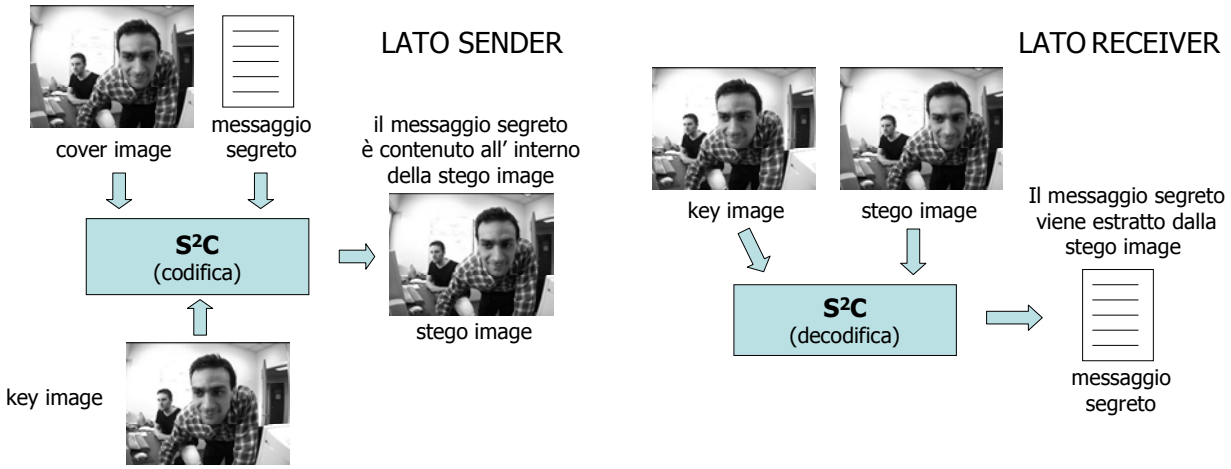


Figura 2 – Sistema steganografico con key image

L'intenzione è quella di unire Stereo visione, Steganografia e Crittografia progettando un nuovo algoritmo che prende il nome di S^2C. Esso deriva dall'acronimo delle tre tecniche di cui sopra (SSC), espresso in "notazione matematica".
S^2C sfrutta la possibilità, suggerita dalla stereo visione, di usare una coppia di immagini (cover e key) per produrre un'immagine stego che sia l'immagine stereo della key image; la mappa di disparità tra stego e key sarà il messaggio segreto.

Infatti, se pensiamo alla key image come ad una delle due immagini stereo che rappresentano il consueto input di un sistema di stereo visione e se il messaggio segreto viene fatto corrispondere alla mappa di disparità tra stego image e key image (si confrontino la figura 3 e la figura 4), allora S^2C risulta essere l'inverso della correlazione (lato sinistro della figura 4).

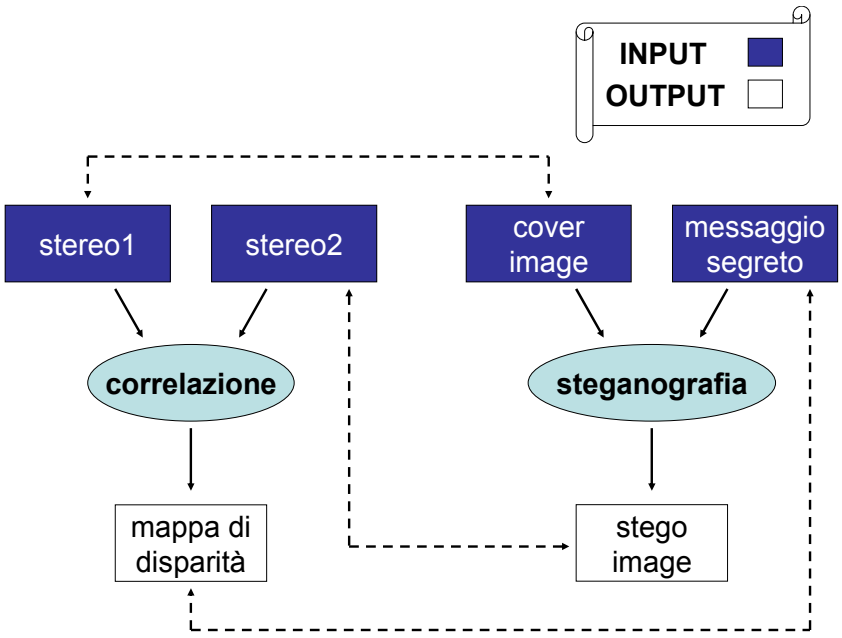


Figura 3 – Mapping tra stereo visione e steganografia

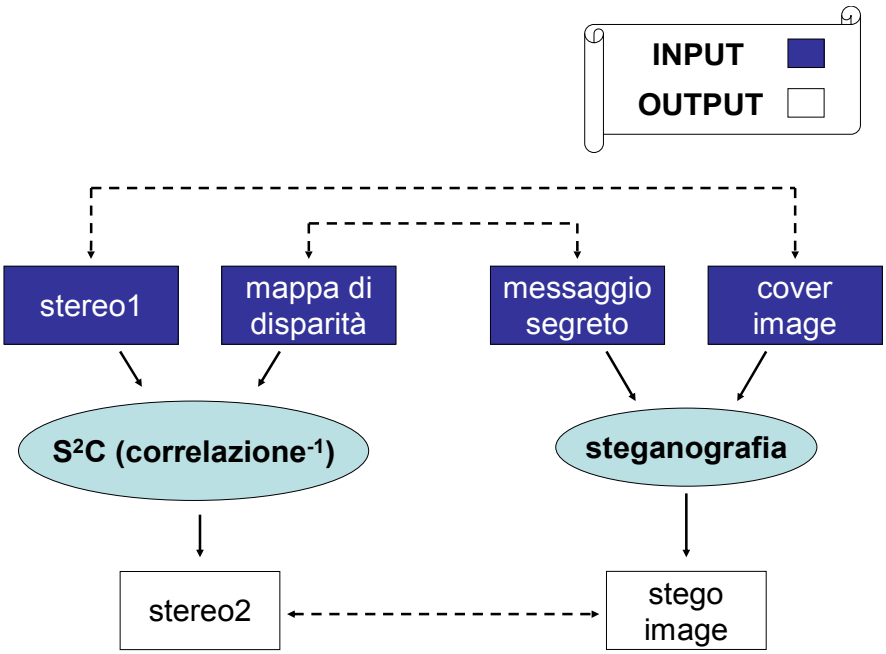


Figura 4 – Corrispondenza tra S^2C e steganografia

La stego image è l'immagine cover modificata in modo tale che il messaggio segreto scaturisca, lato receiver, dal confronto tra essa e la key image (parte destra della figura 2). Le suddette modifiche alla cover image non avvengono in base ai soli

contenuti del messaggio che vi si vuole inserire all'interno (come accade nella steganografia classica), bensì esse scaturiscono, oltre che dai bit del messaggio, anche dal confronto con i dati della key image.
Il risultato originale di S^2C è che la presenza dell'immagine chiave permette, in aggiunta all'occultamento del messaggio (che è la parte strettamente steganografica della procedura), di cifrarlo (con un algoritmo sicuro) dal lato sender e di estrarlo (con estrema facilità) sul lato receiver. In soldoni: **S^2C è al tempo stesso un algoritmo di crittografia ed un algoritmo di steganografia**.
L'inserimento di un nuovo parametro in input (la key image) all'algoritmo classico di steganografia (visto in figura 1), si potrebbe obiettare, appesantisce il processo di inserimento. La presente tesi mostra che, invece, l'uso della key image rende il sistema S^2C più semplice e sicuro dei sistemi steganografici a disposizione oggi.
I risultati dei test comparativi con le tecniche steganografiche note permettono di dire che l'algoritmo S^2C fa propri i principi e le linee guida atti a scongiurare possibili attacchi che emergono dagli studi della steganalisi [19]. In altri termini, la procedura S^2C non è soggetta agli attacchi classici (visuali e statistici [4]) che vengono portati ai più diffusi software per la steganografia.
In più, viene dimostrato che il metodo S^2C permette di cifrare con estrema sicurezza il messaggio che viene fornito ad essa in input.
Lo sviluppo di *S2C Mail*, un software per l'invio di e-mail sicure presentato nel capitolo VI, dimostra, nei fatti, che S^2C può essere usato realmente per inviare contenuti riservati tramite Internet.

Struttura della tesi

Il capitolo I offre una panoramica sulle idee alla base della crittografia e della steganografia. È interessante apprezzare le differenze tra queste due arti, evidenziate in un apposito paragrafo. Questo capitolo, inoltre, definisce con precisione tutti i termini che saranno utilizzati nel proseguo del testo.
Il capitolo II descrive, con dovizia di particolari, le tecniche di stereo visione, con il fine di introdurre l'insieme di conoscenze che rappresenta il fondamento teorico su cui verrà costruito (nel capitolo IV) S^2C.
Il capitolo III illustra i programmi steganografici disponibili in questo momento. Per ognuno di questi software viene mostrato l'attacco che può evidenziare la natura sospetta del suo output.
Il capitolo IV è dedicato alla rigorosa definizione dell'algoritmo S^2C tramite la costruzione, descritta passo dopo passo, di questa inedita tecnica che fonde due scienze da sempre distinte, la steganografia e la crittografia. Inoltre, in questo capitolo viene presentato l'algoritmo S^2C sia nella versione per immagini lossless sia predisposto per immagini lossy.
Il capitolo V dimostra che S^2C cifra il messaggio segreto con un meccanismo equivalente al cifrario di Vernam [22] (e, con qualche accorgimento, al metodo one-time pad), il che garantisce la sicurezza dell'algoritmo di cifratura. La procedura S^2C,

al contempo, nasconde il messaggio segreto all'interno della cover image con prestazioni simili all'algoritmo F5 [35], il che lo rende immune dai più comuni attacchi steganalitici (visuale e statistico) disponibili oggi.
Il capitolo VI conclude questo lavoro, riassumendo quanto è stato proposto in precedenza e mettendo in evidenza i risultati originali conseguiti. In esso sono presenti le descrizioni del software S2C Mail, basato sul paradigma S^2C, per l'invio di e-mail sicure e di una possibile estensione di S^2C anche ai formati video M-JPEG ed MPEG. L'analisi degli sviluppi futuri del presente lavoro conclude il testo.

Capitolo I

Crittografia e Steganografia

Capitolo I

Crittografia e Steganografia

Questo capitolo offre, nell'ordine, una descrizione (veloce) della crittografia ed una più approfondita esposizione dei concetti di base della steganografia. Il contributo critico del capitolo è costituito dall'enunciazione delle differenze tra queste due arti, evidenziate in un apposito paragrafo. Inoltre, il presente contiene le definizioni di tutti i termini che saranno utilizzati nel proseguo del testo ed una sezione dedicata ai moderni utilizzi della steganografia.

I.1 Definizione di crittografia*

La crittografia è la scienza che studia tecniche e metodologie per cifrare (codificare) un testo in chiaro (in inglese *plaintext*), al fine di produrre un testo cifrato (*ciphertext*) comprensibile solo ad un ricevente legittimo (*receiver*), il quale possiede l'informazione sufficiente (detta *chiave*) per decifrarlo, recuperando il testo in chiaro. La crittografia fa parte, insieme alla *crittoanalisi*, della crittologia.
La crittoanalisi studia come *decrittare* un testo cifrato per ottenere il testo in chiaro: il verbo decrittare indica l'azione compiuta da un'entità che non possiede la chiave per recuperare, in modo legittimo, il testo in chiaro. In letteratura, suddetta entità viene indicata con il termine ascoltatore (*eavesdropper*) oppure avversario o anche nemico.

Nota *Al fine di evitare facili fraintendimenti, si precisa che i verbi decrittare e decifrare saranno usati con significati diversi. In questo testo, decifrare indica recuperare il plaintext per mezzo di una chiave conosciuta, decrittare vuole dire tentare di estrarre il testo in chiaro* senza *conoscere la chiave.*

Lo scopo della crittografia è di produrre un messaggio cifrato *m* (che viaggerà poi a partire da un mittente, in inglese *sender*, fino ad un receiver), in modo che nessun avversario sia in grado di decrittare il **contenuto** di *m*. Shannon [33] afferma che un sistema crittografico garantisce la "segretezza perfetta" se la probabilità di ottenere il testo in chiaro rimane la stessa prima e dopo aver esaminato il testo cifrato; al contrario, se il sistema non offre segretezza perfetta, la semplice conoscenza del ciphertext offre informazioni sul plaintext. In letteratura esistono sistemi di cifratura perfetta secondo la definizione di Shannon, tuttavia essi sono stati raramente usati in pratica, poiché devono sottostare a vincoli molto rigidi: ogni chiave deve essere lunga almeno come il messaggio stesso e non può essere adoperata più di una volta (vedi esempio I.1 e seguente teorema I.1).

---

* la quasi totalità di questo paragrafo è tratto da [39]

Esempio I.1 *Un sistema con garanzia di segretezza perfetta è il one-time pad usato per il celebre telefono rosso della Casa Bianca. Lo spazio del plaintext P è* $\{0, 1\}^n$ *, così come lo spazio della chiave k. Lo schema di cifratura-decifratura è simmetrico, con k scelta a caso. Il ciphertext C è prodotto nel seguente modo*

$C = E_k(P) = P \oplus k$ *dove* E_k *indica la codifica (encoding) di P tramite la chiave k,*

mentre il testo in chiaro P si riottiene invertendo il procedimento

$P = D_k(C) = C \oplus k$ *dove* D_k *indica la decodifica di C tramite la chiave k.*

Purtroppo one-time pad necessita di una dimensione dello spazio delle chiavi almeno pari ai bit del messaggio P, quindi se il messaggio è molto lungo k diventa scomoda da gestire. Inoltre, come è stato già detto, k può essere utilizzata una sola volta. □

Teorema I.1 (Shannon) *Un metodo di cifratura non può essere perfetto se lo spazio delle chiavi è più piccolo dello spazio dei possibili messaggi* [18, 22].

I.2 Definizione di steganografia

Il termine steganografia è composto da due parole di origine greca, *stèganos* (che significa nascosto) e *gràfein* (che significa scrivere). Tale termine indica quindi "la scrittura nascosta" o, per meglio dire, l'insieme dei metodi e delle tecniche che permettono a due o più entità (siano esse persone o macchine) di comunicare in modo tale da occultare agli occhi di un eventuale ascoltatore, non tanto il contenuto (come nel caso della crittografia), ma la stessa esistenza di una comunicazione riservata. In altre parole, la steganografia è l'arte di nascondere un messaggio (che si vuole rimanga segreto) all'interno di un'altro messaggio contenitore (che si vuole sia pubblico) di aspetto diverso, spesso totalmente diverso, e **non sospetto**.

I.3 Steganografia nella storia

Già in epoche remote sono state escogitate tecniche per nascondere la presenza di un canale di comunicazione tra parti distinte. Erodoto (486 – 425 AC) racconta la storia di un nobile persiano che fece tagliare a zero i capelli di uno schiavo fidato al fine di poter tatuare un messaggio sul cranio di quest'ultimo; una volta che i capelli furono ricresciuti allo schiavo, questi fu inviato a destinazione, con la sola istruzione di tagliare nuovamente i propri capelli [34].
Gli antichi romani usavano scrivere fra le righe di un testo utilizzando un inchiostro fatto con sostanze naturali come il succo di limone, l'aceto o il latte. Il messaggio nascosto diventava visibile una volta che il testo veniva avvicinato ad una fonte di calore. Questi inchiostri invisibili sono noti anche come inchiostri simpatici [34].
Le griglie di Cardano (1501 – 1626) erano fogli di materiale rigido nei quali venivano ritagliati fori rettangolari a intervalli irregolari; applicando la griglia sopra un foglio

di carta bianca, il messaggio segreto veniva scritto nei buchi (ciascun buco poteva contenere una o più lettere), dopodichè si toglieva la griglia e si cercava di completare la scrittura del resto del foglio in modo da ottenere un messaggio di senso compiuto, il quale poi veniva inviato a destinazione. Applicando sul foglio una copia esatta della griglia originaria, era possibile leggere il messaggio nascosto [32, 34].
Una tecnica molto usata dai nazisti durante la seconda guerra mondiale fu quella dei micropunti fotografici: si tratta di fotografie della dimensione di un punto dattiloscritto che, una volta sviluppate e ingrandite, possono diventare pagine stampate di buona qualità [34].
Sempre durante la seconda guerra mondiale furono impiegate come inchiostri simpatici sostanze molto sofisticate, come ad esempio gli inchiostri al cobalto, che possono essere resi visibili solo mediante l'uso di particolari reagenti chimici.
Un altro modo di nascondere un messaggio è quello di celarlo all'interno di un altro testo. Un acrostico è una poesia - o un testo di qualsiasi tipo - composta intenzionalmente in modo tale che, unendo le prime lettere di ogni capoverso, si ottenga un messaggio di senso compiuto. Esistono numerose varianti di questa semplice idea di base, come il testo che segue (il quale fu realmente inviato da una spia tedesca durante la seconda guerra mondiale):

```
Apparently neutral's protest is thoroughly discounted and ignored.
Isman hard hit. Blockade issue affects pretext for embargo on
by-products, ejecting suets and vegetable oils.
```

Considerando in sequenza la seconda lettera di ogni parola, si ottiene il messaggio:

```
Pershing sails from NY June 1
```
(**Nota**: l'ultima "i" va interpretata come un 1)

Il primo testo stampato di steganografia fu scritto da *Johannes Trithemius* (1462-1516) con il titolo *Steganographia.* Trithemius, nome italianizzato dell'umanista e teologo tedesco Johannes von Heidenberg, detto Tritheim, lo scrisse tra il 1499 e il 1500, dopodichè il libro circolò a lungo in forma manoscritta. Stampato nel 1606, venne poco dopo iscritto nell'*Index Librorum Prohibitorum* in quanto ″pericoloso e colmo di superstizioni″. Una curiosità: la prefazione del libro annuncia in modo provocatorio - anche se oscuro - la presenza di un messaggio nascosto (il dilemma è stato risolto solo nel 1998 da Jim Reeds della AT&T Labs [32]).
Oggi, la steganografia è soprattutto un'arte "digitale", nel senso che il più comune mezzo trasmissivo per i messaggi steganografici è divenuto Internet.
È possibile reperire, senza difficoltà, una vasta gamma di programmi per la modifica di immagini (EzStego per immagini GIF, JSteg, JPHide, OutGuess, F5 per immagini JPEG) o per la trasformazione di brani mp3 (MP3Stego). Infatti, già nel 2001 un articolo de "La repubblica" [23] riportava: *"Negli ultimi due anni il numero di programmi per la steganografia disponibili su Internet - dà l'allarme Neil F. Johnson, un esperto che insegna alla George Mason University - è raddoppiato, arrivando a 140 e oltre"*. Questi fatti indicano un rinnovato interesse per la steganografia, un metodo che, soprattutto dopo l'11 Settembre, vale la pena di studiare (si veda anche il paragrafo I.10).

I.4 Differenza tra crittografia e steganografia

A differenza della crittografia, il cui obiettivo è quello di rendere protetto un dato canale di comunicazione da un eventuale ascoltatore non autorizzato, impedendo a quest'ultimo di accedere al contenuto del canale, le tecniche steganografiche si sforzano di nascondere la mera presenza del canale protetto, utilizzando un canale pubblico (il cui contenuto è accessibile ad un ascoltatore malizioso) come veicolo per lo scambio di messaggi che devono rimanere riservati.
Il fine ultimo della crittografia è quello di proteggere il contenuto di un messaggio, preservando, attraverso un metodo di cifratura, la confidenzialità di tale messaggio. Anche se un testo cifrato è perfettamente riconoscibile come tale (vedi esempio I.2), è fondamentale che un estraneo, che venga in possesso del messaggio, non possa in alcun modo accedere alle informazioni contenute in esso o poterle modificare.
Lo scopo della steganografia, invece, è quello di impedire che un estraneo abbia il seppur minimo indizio che stia avvenendo un trasferimento di dati confidenziali. Nessuno deve poter sospettare che si voglia trasmettere materiale riservato: la steganografia fallisce nel momento in cui la trasmissione viene scoperta, anche se non si è in grado di decrittarne il contenuto.

Esempio I.2* *Il seguente è un messaggio di testo cifrato con il programma Pretty Good Privacy (PGP), probabilmente il più usato e famoso software per l'invio di e-mail criptate al giorno d'oggi.*

```
qANQR1DBwU4D/TlT68XXuiUQCADfj2o4b4aFYBcWumA7hR1Wvz9rbv2BR6WbEUsy
ZBIEFtjyqCd96qF38sp9IQiJIKlNaZfx2GLRWikPZwchUXxB+AA5+lqsG/ELBvRa
c9XefaYpbbAZ6z6LkOQ+eE0XASe7aEEPfdxvZZT37dVyiyxuBBRYNLN8Bphdr2zv
z/9Ak4/OLnLiJRk05/2UNE5Z0a+3lcvITMmfGajvRhkXqocavPOKiin3hv7+Vx88
uLLem2/fQHZhGcQvkqZVqXx8SmNw5gzuvwjV1WHj9muDGBY0MkjiZIRI7azWnoU9
3KCnmpR60VO4rDRAS5uGl9fioSvze+q8XqxubaNsgdKkoD+tB/4u4c4tznLfw1L2
YBS+dzFDw5desMFSo7JkecAS4NB9jAu9K+f7PTAsesCBNETDd49BTOFFTWWavAfE
gLYcPrcn4s3EriUgvL3OzPR4P1chNu6sa3ZJkTBbriDoA3VpnqG3hxqfNyOlqAka
mJJuQ53Ob9ThaFH8YcE/VqUFdw+bQtrAJ6NpjIxi/x0FfOInhC/bBw7pDLXBFNaX
HdlLQRPQdrmnWskKznOSarxq4GjpRTQo4hpCRJJ5aU7tZO9HPTZXFG6iRIT0wa47
AR5nvkEKoIAjW5HaDKiJriuWLdtN4OXecWvxFsjR32ebz76U8aLpAK87GZEyTzBx
dV+lH0hwyT/y1cZQ/E5USePP4oKWF4uqquPee1OPeFMBo4CvuGyhZXD/18Ft/53Y
WIebvdiCqsOoabK3jEfdGExce63zDI0=
=MpRf
```

Il messaggio, oltre ad essere, evidentemente, privo di alcun significato semantico ed a non rispettare alcuna regola sintattica, presenta caratteri che appaiono con un ordine casuale (le occorrenze dei vari caratteri sono equiprobabili), a differenza di qualsiasi messaggio di senso compiuto, in italiano o in inglese o in qualsiasi altra lingua, dove esiste un rapporto di frequenza tra le varie lettere (per esempio, nelle parole della lingua italiana la lettera "a" è molto più frequente della lettera "z"). Per tali motivi, se si considera un ambiente di trasmissione in cui una entità di

---

***** Questo esempio è tratto da [15]

qualsivoglia genere (sia essa di controllo o di disturbo) effettui un esame sul contenuto delle trasmissioni, suddetta entità potrà dedurre con facilità che si tratta di un messaggio criptato o comunque di un messaggio sospetto. In conclusione, il messaggio di cui sopra è un esempio di crittografia, ma ***non è steganografia****. Dal punto di vista della steganografia, la trasmissione, una volta che la presenza di un messaggio nascosto è stata individuata, può considerarsi non riuscita anche se il messaggio nascosto non verrà decrittato; la crittografia, invece, considera fallita una comunicazione quando un estraneo è in grado di ottenere il messaggio originale.* □

I.5 Il problema dei prigionieri

La moderna formulazione degli studi steganografici si deve al lavoro di Simmons [16] che propose il *problema dei prigionieri*. Alice e Bob sono imprigionati in celle separate e desiderano accordarsi su un piano di fuga. Essi hanno la possibilità di scambiarsi dei messaggi, tuttavia tali messaggi vengono controllati dal guardiano Wendy. Se Wendy dovesse accorgersi che i due si stanno scambiando messaggi cifrati, li metterebbe subito in isolamento, impedendo di fatto ogni possibilità di un accordo per fuggire. Alice e Bob devono, pertanto, trovare un modo per occultare il loro piano in un messaggio (ad esempio, un testo o una immagine) che possa apparire innocuo agli occhi di Wendy.
Nella figura I.1 è riportato il modello generale per lo studio della steganografia, nel quale Alice desidera inviare un messaggio segreto *m* a Bob. Per farlo, Alice nasconde *m* in un oggetto di copertura (*cover object*) *c*, ottenendo un oggetto *s* composto dall'unione di *m* e *c* che chiameremo *stego object*. L'oggetto *s* verrà quindi inviato sfruttando un canale pubblico. E' inoltre possibile utilizzare, per costruire lo stego object, anche una chiave segreta *k* (condivisa da Alice e Bob).

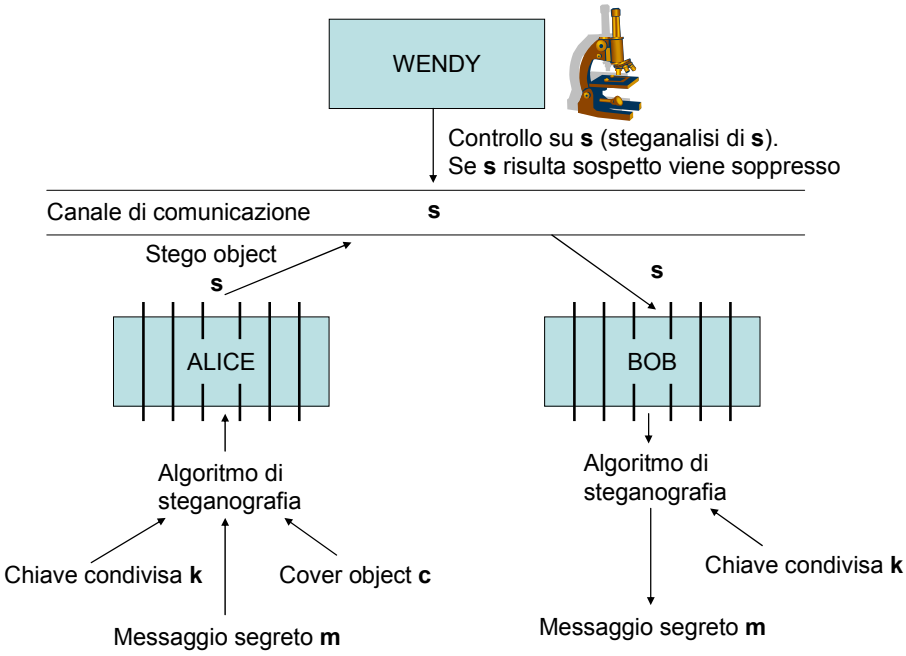


Figura I.1 – Modello generale per la steganografia

Definiamo con maggiore precisione i due termini introdotti sopra: essi rappresentano concetti fondamentali che saranno utilizzati spessissimo nel seguito di questo testo.

Definizione I.2 *Si definisce* cover object *(o cover medium) l'oggetto che sarà usato come contenitore per l'inserimento del messaggio. Diversi oggetti sono stati utilizzati in progetti reali come contenitori per messaggi steganografici: immagini, files audio, video, file systems e pagine html solo per citarne alcuni. Il cover object designa il contenitore originale* prima *che sia effettuata qualsiasi operazione su di esso.*

Definizione I.3 *Si definisce* stego object *(o stego medium) l'oggetto, risultato dell'algoritmo di steganografia (quindi* dopo *che sono state effettuate operazioni su di esso), che contiene (trasporta) al suo interno il messaggio.*

Riepilogando, dati un cover object ed un messaggio, il nostro compito sarà quello di produrre uno stego object che trasporterà al suo interno il messaggio (in figura I.2 il cover medium è un' immagine).

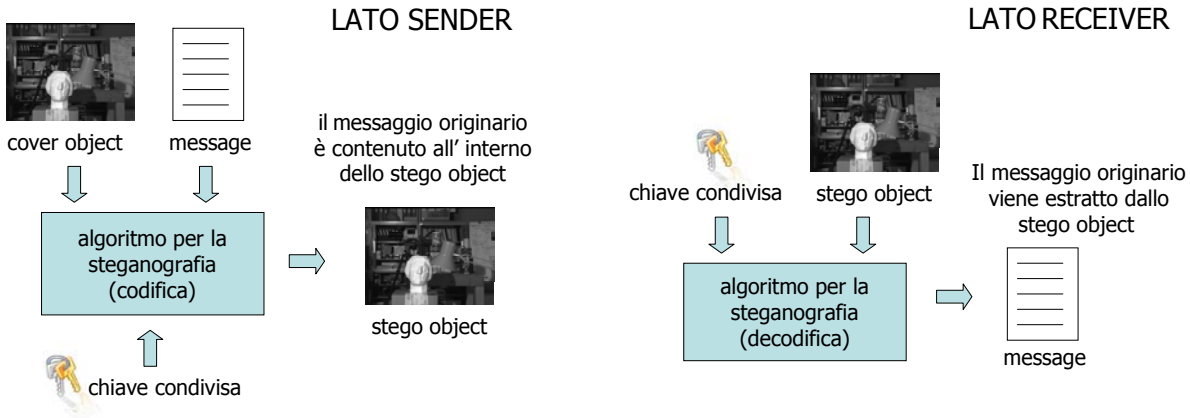


Figura I.2 – Tipico algoritmo steganografico

L'algoritmo per la steganografia può essere sia segreto (cioè condiviso da Alice e Bob ma sconosciuto a Wendy), sia pubblico. In questo testo così come in letteratura, l'algoritmo usato verrà considerato di pubblico dominio, solo la chiave usata per la generazione dello stego-object sarà ritenuta segreta; le precedenti assunzioni (algoritmo pubblico e chiave segreta) corrispondono più o meno al principio di Kerckhoffs proprio della crittografia (si veda il paragrafo I.7).

Esempio I.3 *In un algoritmo che nasconde un messaggio in una immagine, la chiave segreta potrebbe essere un numero da usare come seme per una funzione di generazione pseudo-random. I numeri generati da tale funzione potrebbero rappresentare le locazioni dei pixel che contengono il messaggio [35].* □

Si è detto che Wendy non conosce la chiave segreta che Alice e Bob utilizzeranno, tuttavia è al corrente dell'algoritmo che i due potrebbero, eventualmente, utilizzare per nascondere il loro messaggio. Inoltre Wendy, oltre ad essere libera di esaminare

tutti i messaggi che i due prigionieri Alice e Bob scambiano tra loro, può assumere un ruolo attivo oppure passivo.

Definizione I.4 *Si definisce* passivo *un guardiano che si limiti ad esaminare il messaggio che i due prigionieri si scambiano, cercando di determinare se esso contenga o meno del materiale occultato al suo interno. In caso positivo, il messaggio verrà soppresso ed i due prigionieri saranno puniti. In caso negativo, il messaggio sarà lasciato passare senza intraprendere alcuna azione.*

Definizione I.5 *Si definisce* attivo *un guardiano che possa deliberatamente alterare il messaggio che i due prigionieri si scambiano, anche se non c'è traccia di un possibile contenuto occulto. I cambiamenti che un guardiano attivo può effettuare dipendono dall'algoritmo che si sta adoperando e dal cover object in questione.*

Il compito di un sistema steganografico dovrebbe essere quello di rendere impossibile a Wendy di poter distinguere tra oggetti che non contengono messaggi segreti (cover object) ed oggetti che invece nascondono del materiale (stego object). La steganalisi (si veda il par. I.9) è l'insieme delle tecniche e dei metodi che mirano a fornire strumenti a Wendy per discernere tra cover medium e stego medium. In ogni caso, Wendy deve operare la sua analisi non conoscendo la chiave segreta condivisa da Alice e Bob e, spesso, senza conoscere neppure l'algoritmo che i due stanno utilizzando per nascondere i loro messaggi. Ben si capisce che la steganalisi è di per sé un problema molto complesso. Comunque, solitamente non si richiede a Wendy di scoprire anche il contenuto del messaggio segreto, laddove ella dovesse riscontrarne la presenza; questa assunzione abbassa notevolmente la complessità del problema, rendendo un po' più agevole il compito del guardiano.

I.6 Classificazione delle tecniche*

La steganografia è stata definita (par. I.2) l'arte di nascondere un messaggio (che si vuole rimanga segreto) all'interno di un'altro messaggio contenitore di significato diverso che non risulti sospetto. Analizziamo ora la natura di questo contenitore. Esso può essere costruito a partire da un oggetto già esistente oppure può essere generato ex-novo, utilizzando il messaggio segreto come insieme di vincoli per il processo di generazione. Usualmente, le tecniche steganografiche vengono assegnate a tre classi: la *steganografia sostitutiva*, la *steganografia selettiva* e la *steganografia costruttiva*.

I.6.1 Steganografia sostitutiva

Le tecniche sostitutive sono le più utilizzate ed hanno una diffusione tale che spesso, con il termine steganografia, si fa implicito riferimento ad esse. Queste tecniche sfruttano i bit ridondanti presenti nel cover object, dove con ridondanti si intendono quei

* gran parte di questo paragrafo è tratto da [21]

bit che possono essere sostituiti senza compromettere l'integrità dell'oggetto. Ad esempio, il rumore presente in un canale di comunicazione (trasmissioni telefoniche, brani musicali, etc…) può essere sostituito da un altro segnale, che rappresenta il messaggio segreto, senza introdurre una perdita di informazione sui contenuti trasportati da quel canale. Si tende, solitamente, a considerare ridondante il bit meno significativo di una immagine, anche se purtroppo questa assunzione non è vera in generale [19].

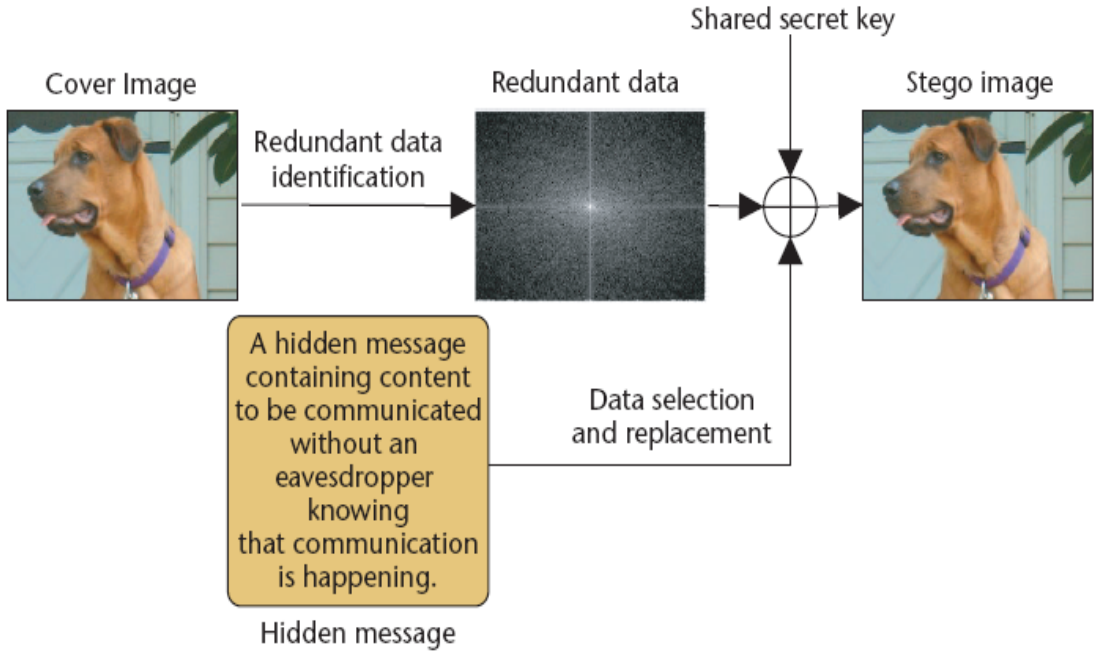


Figura I.3 – Steganografia sostitutiva, tratto da [4]

Nel capitolo III sono riportate le tecniche attualmente più usate per la sostituzione dei bit ridondanti. Esse consistono nel sostituire un elemento di scarsa importanza (in certi casi di importanza nulla) da file di vario tipo, con il messaggio segreto che si vuole nascondere. Il principale difetto di queste tecniche risiede nella possibile alterazione delle caratteristiche statistiche del rumore presente nel contenitore. Ad esempio, la sostituzione operata può modificare il rapporto tra le frequenze dei vari colori in un'immagine [4, 19]. Un ascoltatore potrebbe disporre di un modello del rumore adatto al particolare tipo di file contenitore che ha intercettato ed utilizzare tale modello per analizzarlo: se il rumore presente nel file non è conforme al modello, allora quel file è da considerarsi sospetto. In generale, questo tipo di attacco non è per niente facile da realizzare, data l'impossibilità pratica di costruire un modello che tenga conto di tutte le possibili sorgenti di errori/rumori; tuttavia, in proposito, esistono degli studi specifici (ad esempio, per i file JPEG [19, 27, 28, 29, 38]) che hanno avuto discreto successo (cfr. III.9).
La steganografia selettiva e quella costruttiva hanno proprio lo scopo di eliminare questo difetto della steganografia sostitutiva.

I.6.2 Steganografia selettiva

La steganografia selettiva ha valore puramente teorico e, per quanto se ne sappia, non viene realmente utilizzata nella pratica. L'idea su cui si basa è quella di procedere per tentativi, ripetendo una stessa misura fintanto che il risultato non soddisfa una certa condizione. Facciamo un esempio per chiarire meglio. Si fissi una funzione *hash* semplice da applicare ad un'immagine in forma digitale (una funzione hash è una qualsiasi funzione definita in modo da dare risultati ben distribuiti nell'insieme dei valori possibili; tipicamente questo si ottiene decomponendo e mescolando in qualche modo le componenti dell'argomento); per semplificare al massimo, assumiamo che la funzione valga 1 se il numero di bit uguali a 1 del file che rappresenta l'immagine è pari, altrimenti valga 0 (si tratta di una assunzione davvero poco realistica ma, come dicevamo, questa discussione ha valore esclusivamente teorico). Così, se vogliamo codificare il bit 0 procediamo a generare un'immagine con uno scanner; se il numero di bit dell'immagine uguali a 1 è dispari ripetiamo di nuovo la generazione, e continuiamo così finché non si verifica la condizione opposta. Il punto cruciale è che l'immagine ottenuta con questo metodo contiene effettivamente l'informazione segreta, ma si tratta di un'immagine "naturale", cioè generata dallo scanner senza subire alcuna manipolazione successiva. L'immagine è semplicemente sopravvissuta a un processo di selezione (da cui il nome della tecnica), quindi non si può dire in alcun modo che le caratteristiche statistiche del rumore presentino una distorsione rispetto ad un modello di riferimento. Come è evidente, il problema di questa tecnica è che è troppo dispendiosa rispetto alla scarsa quantità di informazione che è possibile nascondere.

I.6.3 Steganografia costruttiva

La steganografia costruttiva affronta lo stesso problema nel modo più diretto, tentando di sostituire il rumore presente nel medium utilizzato con l'informazione segreta opportunamente modificata in modo da imitare le caratteristiche statistiche del rumore originale. Secondo questa concezione, un buon sistema steganografico dovrebbe basarsi su un modello del rumore e adattare i parametri dei suoi algoritmi di codifica in modo tale che il falso rumore contenente il messaggio segreto sia il più possibile conforme al modello. Questo approccio è senza dubbio valido, ma presenta anche alcuni svantaggi. Innanzitutto, non è facile costruire un modello del rumore: la costruzione di un modello del genere richiede grossi sforzi ed è probabile che qualcuno, in grado di disporre di maggior tempo oppure di risorse migliori, riesca a costruire un modello più accurato, riuscendo ancora a distinguere tra il rumore originale ed un suo sostituto. Inoltre, se il modello del rumore utilizzato dal metodo steganografico dovesse cadere nelle mani del nemico, questi potrebbe analizzarlo per cercarne possibili difetti e quindi utilizzare proprio il modello stesso per controllare che un messaggio sia conforme ad esso. Così, il modello, che è parte integrante del sistema steganografico, fornirebbe involontariamente un metodo di attacco particolarmente efficace proprio contro il sistema stesso.

I.7 Il principio di Kerckhoffs

Il principio di Kerckhoffs [20, 22] è stato sviluppato nel campo della crittografia e, semplificando al massimo, può essere formulato nel modo seguente:

Principio di Kerckhoffs *Quando due parti desiderano comunicare con riservatezza usando un sistema crittografico, la sola cosa che esse devono mantenere segreta è la chiave che intendono utilizzare.*

Inoltre, riportiamo la definizione di schema crittografico attaccabile con successo (in inglese *breakable encryption scheme*).

Definizione I.6 *Uno schema per la cifratura si dice* attaccabile con successo *(breakable) se una terza parte, senza una conoscenza a priori della chiave condivisa, può* sistematicamente *ottenere il testo originale a partire dal messaggio cifrato.*

Per completezza si espongono tutti i requisiti (6 punti noti in letteratura come *Kerckhoffs' desiderata*) che dovrebbe rispettare un sistema di cifratura.

1. Il sistema dovrebbe essere, se non teoricamente, impossibile da rompere almeno in pratica.

2. La pubblicazione dei dettagli del sistema non dovrebbe compromettere la sicurezza della comunicazione.

3. La chiave dovrebbe essere facilmente memorizzabile (senza che sia necessario annotarla) e facilmente sostituibile.

4. Il messaggio cifrato dovrebbe poter essere trasmesso con un telegrafo.

5. L'apparato per le operazioni di cifratura dovrebbe essere trasportabile ed utilizzabile da una singola persona.

6. Il sistema dovrebbe essere semplice e non richiedere la conoscenza di una lunga lista di regole né sforzo mentale.

Questo insieme di requisiti fu redatto nel 1883 e per buona parte risulta valido ancora oggi. In particolare, il secondo punto afferma che le trasformazioni operate dall'algoritmo di cifratura possano essere rese note e che la sicurezza del sistema dovrebbe risiedere solo nella chiave scelta. Anche l'ultimo punto riveste una certa importanza: la semplicità deve essere uno dei requisiti base per un sistema crittografico, così come per un sistema steganografico. Infatti, un sistema semplice è anche implementabile con la massima efficienza.

I.8 Steganografia su immagini

Si è detto nel paragrafo I.5 che diversi tipi di file possono essere utilizzati come contenitori per messaggi steganografici: immagini, files audio, video, file systems e pagine html. Per esempio, malelingue sostengono che negli anni ’80 l’ex primo ministro inglese Margaret Thatcher, preoccupata per la fuga di notizie riservate, lasciate filtrare alla stampa da suoi non troppo fedeli collaboratori, fece programmare i loro word processor in modo che il nome dello scrivente fosse codificato nella spaziatura delle parole [25].
Se esiste una gamma così vasta di possibili contenitori, perché usare proprio le immagini digitali e non brani musicali o pagine html? Proviamo a rispondere.
La steganografia (si consideri solo quella sostitutiva, dato che le altre tipologie sono praticamente inutilizzate) si basa sulla sostituzione delle informazioni ridondanti presenti nel cover object con i dati del messaggio segreto; si ottiene così lo stego object che differisce solo leggermente dal cover. Se Alice e Bob scelgono come contenitore un file facilmente reperibile (come loghi di aziende, foto di attori famosi, hit musicali, pagine web di quotidiani), anche Wendy potrebbe fare lo stesso. Quest’ultima, in possesso del contenitore, non avrà difficoltà nel trovare le differenze tra il cover e lo stego object che Alice e Bob tentano di scambiarsi. Wendy, trovate queste differenze (se esiste un messaggio nascosto debbono per forza esistere delle discrepanze), etichetta come sospetto lo stego medium e sopprime la comunicazione.

1. Alice invia a Bob uno stego object ottenuto da un noto logo reperibile sul web.
2. Wendy prende in consegna lo stego object per consegnarlo a Bob.

   Anche Wendy conosce il logo e decide di scaricarne una copia dal sito ufficiale.
3. Wendy scopre che lo stego object non corrisponde alla copia scaricata e si insospettisce.
4. Wendy sopprime lo stego object.

Figura I.4 – Mai scegliere un cover-object facilmente reperibile

La steganografia ha, perciò, la necessità di utilizzare cover object che siano possibilmente inediti e mai usati in precedenza: non c’è niente di più facile che scattare una foto, per ottenere un cover con queste caratteristiche. Sembra, infatti, improbabile comporre di volta in volta nuovi brani musicali per inviare messaggi

(essi risulterebbero comunque sospetti) oppure modificare semplicemente il testo di pagine html (in tal modo si produrrebbero errori ortografici e frasi senza senso) ovvero, più realisticamente, utilizzare gli spazi in file testuali o alcuni tag dei file html [24] (un siffatto modello di rumore può essere replicato se si conosce l'algoritmo di steganografia adoperato).
Le immagini, soprattutto quelle compresse, contengono del rumore difficilmente modellabile, dal momento che la compressione usata per alleggerirli rende più complicato capire se ci sono state distorsioni di altro tipo; in più, il web rappresenta un comodo canale di trasmissione per esse (difatti, molti sono i siti che consentono di pubblicare immagini in maniera anonima, tra cui la maggior parte di quelli pornografici e quello di aste eBay [4]) e l'enorme diffusione delle immagini, sia in Internet sia come allegati in e-mail, le rende difficilmente sospettabili e poco realisticamente controllabili singolarmente.
Se si hanno particolari esigenze di sicurezza, esiste la possibilità di scegliere cover object con una capacità molto superiore alle proprie esigenze. Distribuire i bit del messaggio segreto su un volume maggiore di dati permette di diluire i mutamenti da effettuare, rendendo praticamente impossibile compiere un attacco al sistema steganografico. Le immagini si prestano naturalmente a questo fine: invece di utilizzare i bit meno significativi di tutti i pixel di un'immagine, si può scegliere di modificarne solo uno ogni venti.
In conclusione, le immagine compresse sono il miglior tipo di cover, oggi, per effettuare trasmissioni steganografiche.

I.9 La Steganalisi

La steganalisi è l'insieme dei metodi e delle tecniche capaci di attaccare un sistema steganografico. Così come la crittoanalisi ha il compito di svelare l'output cifrato di un sistema crittografico, lo scopo della steganalisi è quello di etichettare con certezza un oggetto come sospetto (contenente, cioè, al suo interno un messaggio occulto) o come innocuo (privo di informazioni nascoste).
In altre parole, *"dove nella crittoanalisi il fine è quello di rivelare il dato, nella steganalisi il fine è quello di scoprirne la presenza"* [26].
Fare steganalisi è intrinsecamente un compito complesso poiché, spesso, non si è in possesso di tutte le informazioni necessarie per compiere un esame che possa portare ad esiti certi. Si tratta, molte volte, di indovinare i passi che avrebbe potuto compiere il sender del presunto messaggio steganografico, dove con indovinare si intende cercare il cammino più probabile scelto dal sender nell'insieme delle mosse possibili.
In letteratura, si suole suddividere le varie procedure di attacco steganalitico specificando il tipo di informazioni che si posseggono.

1. **Attacco stego-only**. Si possiedono solo i dati da analizzare, cioè si può analizzare solo lo stego object. In questo caso si può assumere un modello di rumore plausibile per il tipo di stego object in esame tentando di ottenere una

percentuale di falsi positivi medi. Si tratta dell'avvenimento più comune, oltre che di quello con minore informazione possibile.

2. **Attacco known-cover**. Si posseggono il dato originale ed il dato alterato. È il caso riassunto in figura I.4, dove Wendy possiede sia lo stego object sia il cover object. Si tratta dell'eventualità più fortunata, perché analizzando le differenze trai due, non è difficile intuire l'algoritmo di steganografia che è stato utilizzato.

3. **Attacco known-message**. Si possiedono sia il messaggio segreto sia lo stego object. Analizzando il rapporto tra i bit del messaggio e quelli del contenitore possono essere scovate le relazioni che li legano. Le stesse relazioni possono essere ricercate in altri stego object per verificarne la natura.

4. **Attacco known-decoding (chosen stego)**. Si conosce l'algoritmo usato per estrarre il messaggio segreto. Teoricamente, è possibile utilizzare un attacco *brute force* per estrarre il messaggio dallo stego medium.

5. **Attacco known-encoding (chosen message)**. Si è in possesso dell'algoritmo usato per inserire il messaggio segreto. È possibile studiare gli effetti che producono vari messaggi su potenziali cover, semplicemente realizzando un sufficiente numero di nuovi stego object di prova. Se questi stessi effetti vengono riscontrati sullo stego medium che era in esame, quest'ultimo risulterà sospetto.

Particolare attenzione va riposta nel caso 2, poiché lo stego object che si intercetta potrebbe essere la versione compressa del cover. Esponiamo uno scenario molto semplificato, ma che potrebbe realmente verificarsi.

Esempio I.4 *Alice scarica dal sito dell'Inter una foto di Luis Figo in formato lossless e poi la trasforma in un file con codifica lossy. Alice, quindi, invia il file lossy a Bob,* ***senza*** *inserire alcun messaggio segreto. Wendy, che conosce il mondo del calcio, al momento del controllo, scarica anch'essa la foto originale (lossless) di Figo dal sito dell'Inter, al fine di confrontare le due immagini: quella spedita da Alice e quella appena scaricata. Poiché Wendy non sa che la foto originale è stata compressa (con perdita di informazione) da Alice, trova delle discrepanze tra le due e conclude, errando, che la comunicazione è steganografica.* □

Il precedente esempio, nella sua banalità, mette in evidenza quanto sia difficile analizzare uno stego medium. Infatti, qualsiasi immagine che sia il risultato di una conversione analogico/digitale (come le immagini scattate con una macchinetta digitale), così come qualsiasi immagine compressa, contiene del rumore molto difficile da predire poiché legato ad una moltitudine di variabili (se la macchina

fotografica è di scarsa qualità potrebbe inserire nell'output disturbi non ipotizzabili da nessun modello del rumore generale).
Al momento esistono diverse tecniche steganalitiche applicate con successo a diversi software steganografici disponibili in commercio [4, 27, 28, 29]. Esse verranno presentate, con dovizia di particolari, nel capitolo III che tratta proprio gli approcci già noti alla steganografia sulle immagini.

I.10 Utilizzi della steganografia

La steganografia al giorno d'oggi è utilizzata da molte organizzazioni militari e di intelligence per effettuare comunicazioni riservate. Inoltre, può essere usata laddove non sia consentito l'uso della crittografia. Infatti, come risulta chiaro dall'esempio I.2, è molto semplice per una entità di controllo sulle comunicazioni (come potrebbe essere un regime dittatoriale o repressivo) intercettare materiale cifrato e bloccarne l'invio.
Nel campo del commercio elettronico, la steganografia potrebbe essere utilizzata nell'implementazione di sistemi sicuri di pagamento on-line, in aggiunta ai normali protocolli crittografici come SSL (Secure Socket Layer) e SET (Secure Electronic Transactions).
Potrebbe, altresì, essere utilizzata per l'invio di informazioni di decodifica in un sistema di codifica/decodifica per canali televisivi a pagamento.
Un sistema di passaporti elettronici (o comunque di qualsiasi tipo di documenti) può servirsi di foto con informazioni nascoste, utilizzabili dalle forze di polizia per aumentare la sicurezza o per segnalare, a loro insaputa, i malviventi.
In ambito medico, la realizzazione di radiografie che celino al proprio interno i dati sensibili del paziente potrebbe risolvere elegantemente i problemi legati alla privacy.
Un sistema di steganografia sarebbe (ed è, si veda S2C Mail nel capitolo VI) inglobabile, con estrema facilità, in qualsiasi client di posta così come in qualsivoglia browser web.
Purtroppo, la steganografia si presta anche a scopi tutt'altro che benemeriti. I primi allarmi pubblici per gli usi terroristici della steganografia sono apparsi sulla stampa statunitense all'inizio del 2001 [30, 31]. Questi articoli riportavano come i terroristi stessero usando la steganografia per sottrarre le loro comunicazioni ai nuovi, e più stringenti, controlli previsti dalle leggi USA. Le pratiche steganografiche, sempre secondo questi articoli, sarebbero consistite nell'occultamento di messaggi all'interno di immagini pubblicate in Internet durante aste on-line (il sito prediletto sarebbe stato quello di eBay). Sebbene non venisse riportata alcuna informazione tecnica che potesse supportare questi allarmi, i summenzionati articoli sono stati oggetto di grande attenzione ed hanno prodotto un'eco notevole.
Esistono, in ogni modo, preoccupanti indizi a supporto dell'uso terroristico della steganografia. *"C'è la pista di Jamal Beghal, il leader del commando che stava preparando un attentato all'ambasciata americana di Parigi: l'uomo, addestrato in Afghanistan dove aveva incontrato un luogotenente di Bin Laden, aveva istruito il*

suo gruppo affinché tutte le comunicazioni interne fossero fatte attraverso immagini "ritoccate" pubblicate sulla Rete” [23].

Nota *Così come un coltello da cucina può tagliare il pane ma può anche ferire, la steganografia può essere sia usata per scopi apprezzabili (come la tutela della privacy oppure la difesa di materiale sottoposto a copyright) sia per fini deprecabili (organizzazione di piani criminosi, invio di immagini pedo-pornografiche contenute in immagini dall’aspetto innocuo).*
Il presente lavoro, nel capitolo conclusivo, illustra una applicazione (S2C Mail) per l’invio di e-mail che implementa le idee che si trovano nei capitoli IV e V. Usare S2C Mail è un modo per mantenere la segretezza della quale si può sentire la necessità. Quale sia questa necessità dipende solo dall’utilizzatore dell’applicazione.

Capitolo II

La Stereo Visione

Capitolo II

La Stereo Visione

Questo capitolo presenta la stereo visione come mezzo per l'ottenimento di una mappa di profondità a partire da due immagini: si tratta del fine classico della stereo visione, che viene impiegata sia per la visione artificiale sia per la realizzazione di carte geografiche tridimensionali a partire da foto aeree.
L'uso che si fa in questo testo della stereo visione non è il normale impiego che ne fa l'intelligenza artificiale, bensì le tecniche di stereo visione verranno adoperate, come diverrà chiaro nel capitolo IV, in maniera innovativa.

II.1 Definizione di stereo visione

La stereo visione è una tecnica di ottica inversa consistente nell'ottenere informazione di profondità da una coppia di immagini, provenienti da due telecamere che inquadrano una scena da differenti posizioni. Una singola locazione fisica nello spazio tridimensionale della scena osservata corrisponde ad una **unica** coppia di pixels nelle due immagini. Questo è il principio fondante della stereo visione.

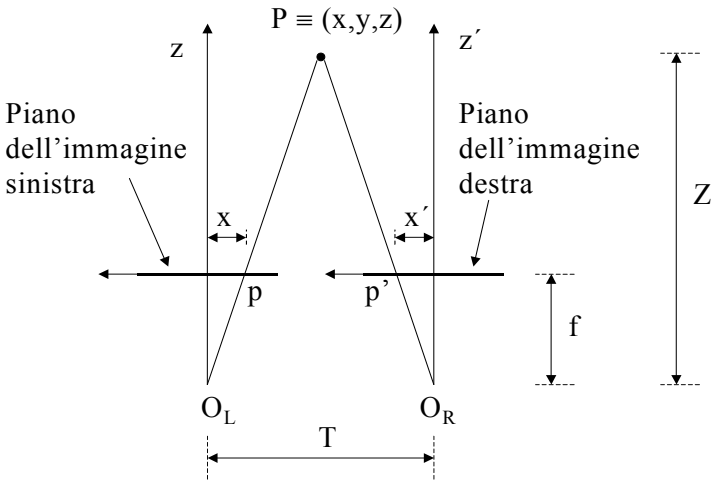


Figura II.1 – Geometria stereo per due telecamere

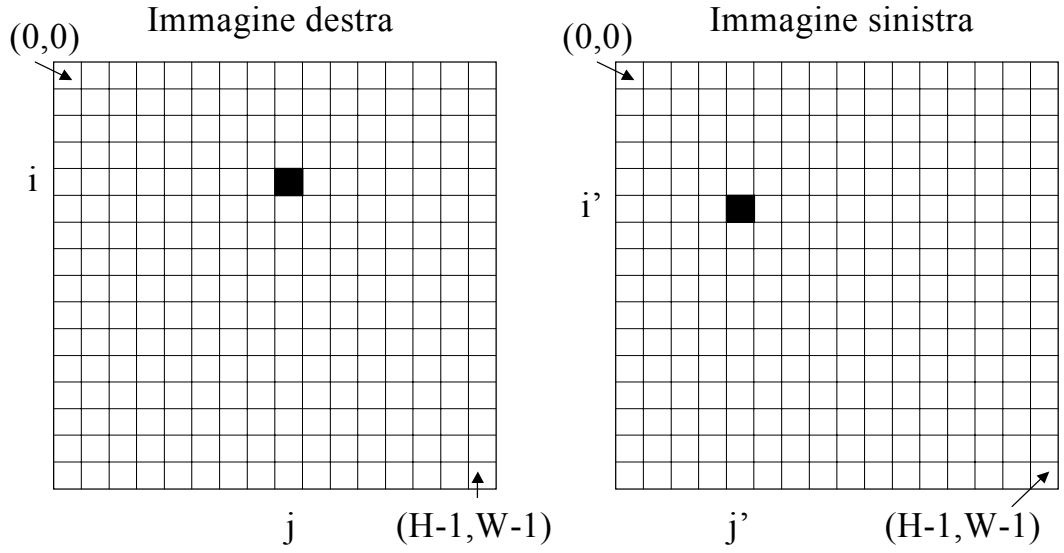
Avendo a disposizione le due immagini prodotte dalle telecamere, se è possibile localizzare la coppia di pixels [p, p'] che corrisponde ad un stesso punto P dello spazio 3D, allora (si veda la figura II.1) è possibile determinate le coordinate *(x, y, z)* di P. Definiamo p e p' una coppia coniugata, con p coniugato di p' e viceversa.

Definizione II.1 *Una coppia coniugata è una coppia di punti in immagini diverse che sono proiezione dello stesso punto della scena.*

In generale, noti gli accoppiamenti tra i punti delle due immagini e nota la posizione reciproca delle telecamere ed i parametri interni del sensore è possibile ricostruire la posizione nella scena dei punti che sono proiettati sulle due immagini.
Si considerino le immagini sinistra e destra come matrici di pixels (la seguente figura II.2 ne darà un'idea): *p* e *p′* possono essere caratterizzati in base ai rispettivi valori di riga e di colonna. Il pixel *p* avrà coordinate riga-colonna *(i, j)* mentre *p′* si troverà in posizione *(i′, j′)*.

Definizione II.2 *La disparità è la differenza (vettore) tra punti (pixels) di una coppia coniugata quando le due immagini sono sovrapposte.*

La disparità tra *p* e *p′* sarà la distanza tra le locazioni *(i, j)* e *(i′, j′)*. Questa distanza può essere quantificata mediante un intero che stia ad indicare la differenza in pixels tra le due posizioni della coppia coniugata.



H = numero di righe (altezza dell'immagine)

W = numero di colonne (ampiezza dell'immagine)

Figura II.2 – Le immagini possono pensarsi come matrici di pixels

Definizione II.3 *La mappa di disparità è l'insieme dei valori di disparità calcolati* per ogni punto *dell' immagine di riferimento (per esempio, quella sinistra).*

Chiaramente, la disparità può essere calcolata solo per punti della scena che sono visibili in entrambe le immagini; un punto visibile in una immagine ma non all'altra viene detto essere *occluso*.

II.2 Le occlusioni

Il mondo è pieno di occlusioni. In ogni scena, con un po' di attenzione, è possibile trovare decine di zone di occlusione.
Si osservi la sottostante figura II.3. Il punto P_V è visibile da entrambe le telecamere, mentre il punto P_O è visibile solo attraverso l'obiettivo O_L; di conseguenza, mentre la profondità del punto P_V può essere calcolata, la disparità di P_O non è calcolabile, a meno che non si aggiungano immagini addizionali della scena o si facciano assunzioni sulla geometria di quest'ultima.

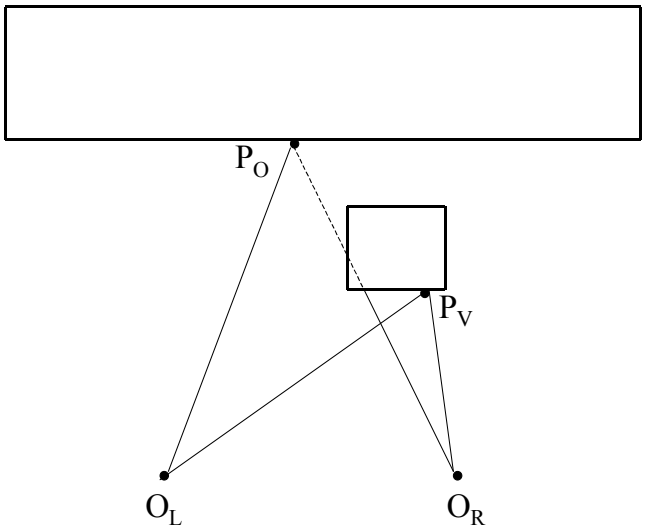


Figura II.3 – Il punto P_O è occluso per O_R

Il lettore provi, ora, ad effettuare una semplice esperienza: guardando di fronte a sè provi a coprire il proprio occhio destro con una mano. Una parte (quella all'estrema destra) della scena scomparirà alla vista. Ora, provi a coprire l'occhio sinistro: questa volta a scomparire sarà la parte più a sinistra della scena. Le due parti (sull'estrema sinistra e sull'estrema destra) che scompaiono sono alcune delle zone di occlusione di cui soffrirebbe la scena se venisse ripresa da due telecamere i cui centri ottici fossero posizionati al posto degli occhi del lettore. Questo è un primo tipo di occlusione (occlusione laterale), inevitabile poiché dovuto allo sfasamento dei due obiettivi.
Esiste, infatti, un secondo tipo di occlusione dovuto alla posizione degli oggetti nella scena: l'occlusione di secondo piano. Quest'ultima si verifica quando la presenza di oggetti molto vicini alle telecamere nasconde ad un obiettivo, ma non all'altro, parte degli elementi della scena che si trovano in secondo piano. Si veda a tal proposito la figura II.4 nella pagina seguente.

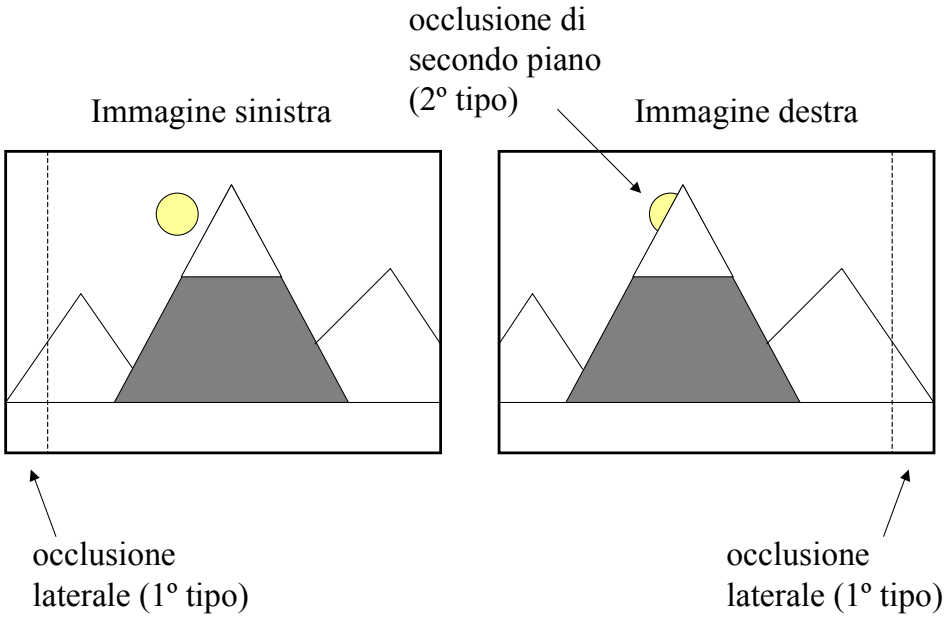


Figura II.4 – I tipi di occlusione

II.3 La ricostruzione della profondità

Il processo di ricostruzione della struttura tridimensionale della scena deve confrontarsi con la soluzione di tre principali problemi: la *calibrazione*, la *corrispondenza* e la *ricostruzione*.

II.3.1 La Calibrazione

Il problema della calibrazione concerne la definizione della geometria interna ed esterna di un sistema di telecamere. La geometria esterna (parametri estrinseci) si riferisce alla posizione reciproca e all'orientazione relativa di ogni camera all'interno del sistema. La geometria interna (parametri intrinseci) riguarda la distanza focale, il centro ottico e la distorsione delle lenti per ogni singola telecamera del sistema. Una stima accurata di questi parametri è necessaria se si vuole mettere in relazione l'informazione che deriva dall'immagine (espressa in pixel) ottenuta dalle camere del sistema con l'effettiva posizione ed orientazione delle telecamere nello spazio. L'idea è che, conoscendo le corrispondenze tra punti 3D di coordinate assolute note e le loro proiezioni, sia possibile ottenere i parametri incogniti risolvendo le equazioni della proiezione prospettica.

II.3.2 La Corrispondenza

Il problema della corrispondenza consiste nel trovare per ogni punto della prima immagine il corrispondente punto nella seconda, per poi calcolare la disparità tra

questi due punti e produrre la mappa di disparità. Il calcolo dell'accoppiamento è possibile sfruttando il fatto che le due immagini differiscono solo lievemente, sicché un particolare della scena appare simile nelle due immagini. Provare tutte le possibili corrispondenze è fuori dalle capacità degli attuali processori, a causa della complessità esponenziale del problema; per tale motivo, si è tentato di introdurre dei vincoli che lo rendano trattabile. Questi vincoli sono principalmente di tre tipi:

1) Vincoli geometrici imposti dal sistema di cattura delle immagini: probabilmente il più importante di questi vincoli è il *vincolo epipolare*, grazie al quale il problema di ricerca delle corrispondenze, in origine bi-dimensionale, può essere ridotto ad uno mono-dimensionale.

2) Vincoli geometrici derivanti dall'ipotesi di continuità a tratti delle superfici nella scena: il fatto che il mondo solitamente è continuo a tratti implica che punti vicini di una scena abbiano valori di profondità simili, quindi di disparità.

3) Vincoli fisici come quelli derivanti dalla interazione tra oggetti e luminosità della scena. Il modello più semplice e maggiormente usato è il modello *Lambertiano* [paragrafo II.5].

E' bene sottolineare che, anche utilizzando questi vincoli, l'impossibilità di ovviare ad alcuni accoppiamenti ambigui generati da occlusioni, errori nella texture e specularità delle immagini impedisce di definire soluzioni generali applicabili al problema.

II.3.3 La Triangolazione

Il problema della triangolazione consiste nel determinare la struttura tridimensionale di una scena a partire dalla mappa di disparità. La profondità di un punto P nello spazio, ripreso da due telecamere con centri ottici O_L ed O_R, è definita dall'intersezione di due rette. La prima di queste rette è quella che congiunge il centro ottico O_L della telecamera di sinistra con il punto p proiezione di P sull'immagine sinistra; la seconda retta unisce il centro ottico O_R della camera di destra con il punto p' proiezione di P sull'immagine destra (si riprenda la figura II.1). Data la distanza T (chiamata *baseline*) tra O_L ed O_R, e nota la distanza focale f delle telecamere, la profondità del punto P può essere calcolata sfruttando le leggi sulla similarità dei triangoli

$$N = f\frac{T}{d} \tag{2.1}$$

dove d è la disparità $d = x - x'$ trovata per P, convertita in unità metriche. Bisogna, però, effettuare una precisazione. Calcolare la disparità attraverso la semplice differenza $x - x'$ dei valori di ascissa dei punti proiezione p e p' è possibile solo se si effettua una assunzione. L'assunzione in questione è che alle immagini sia possibile

applicare il già citato vincolo epipolare, il quale afferma che il corrispondente di un punto in una immagine può trovarsi solo sulla linea epipolare propria del punto nell'altra immagine.

II.4 La Geometria Epipolare

Lo sfruttamento della geometria epipolare ci aiuta nel ridurre lo spazio di ricerca delle corrispondenze: dato un punto m_1 nella immagine 1, si vuole cercare il suo coniugato m_2 nella immagine 2. Alcune semplici considerazioni geometriche indicano che il punto coniugato di m_1 deve giacere su di una linea retta nella immagine 2, chiamata *linea epipolare* di m_1.

Figura II.5 – Geometria epipolare applicata ad immagini stereo

Se le matrici di trasformazione prospettica sono note (tramite la calibrazione) è possibile ottenere l'equazione delle linee epipolari, riducendo così il problema della corrispondenza ad una ricerca mono-dimensionale.

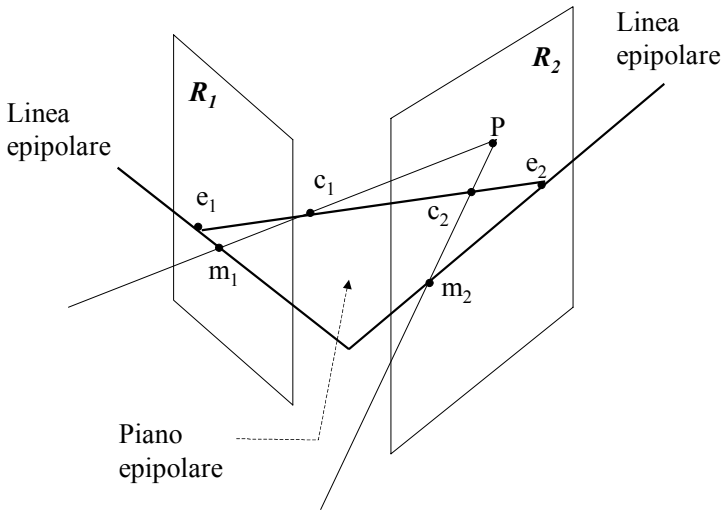


Figura II.6 – Determinazione del piano epipolare

Si consideri il caso illustrato in figura II.6: dato un punto m_1 nell'immagine 1, il suo coniugato m_2 nell'immagine 2 è vincolato a giacere sull'intersezione del piano immagine con il piano determinato da m_1, c_1 e c_2, detto piano epipolare. Questo avviene poiché il punto m_2 può essere la proiezione di un qualsiasi punto nello spazio giacente sul raggio ottico di m_1. Inoltre, si osserva che tutte le linee epipolari di una immagine passano per uno stesso punto, chiamato *epipolo*, e che i piani epipolari costituiscono un fascio di piani che hanno in comune la retta passante per i centri ottici c_1 e c_2.

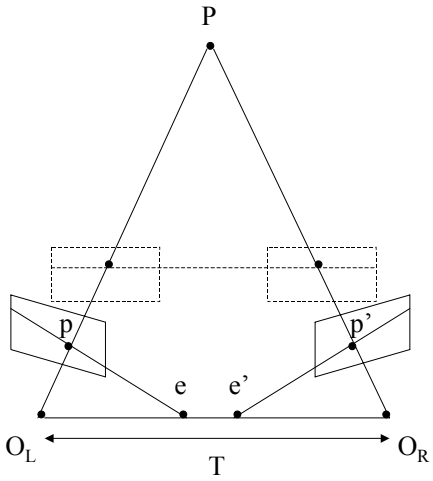


Figura II.7 – Due arbitrarie immagini della stessa scena possono essere rettificate lungo le linee epipolari (in neretto) per produrre scan lines allineate (tratteggiate)

Il punto P in figura II.7 è rappresentato dalle telecamere destra e sinistra rispettivamente come i punti p e p'. La baseline T ed i raggi ottici O_LP e O_RP definiscono il piano di proiezione per il punto P, ovvero il piano epipolare. La linea epipolare che passa attraverso il punto p' è l'immagine del raggio opposto, O_LP, che passa attraverso p. Il punto in cui le linee epipolari di una immagine intersecano la baseline è l'epipolo (e è l'epipolo per p, e' per p'). L'epipolo e corrisponde al centro ottico O_R visto dalla telecamera di sinistra.

II.5 Il modello Lambertiano

La luminosità di un pixel p nell'immagine è proporzionale alla quantità di luce diretta verso il sensore visivo dal pezzo di superficie S_p che proietta il pixel p. Questo dipende a sua volta dalle proprietà di riflettanza di S_p , dalla posizione e distribuzione delle sorgenti di luce. Vi è anche una dipendenza dalle proprietà di riflettanza del resto della scena, infatti le altre superfici della scena possono servire da sorgenti di luce indiretta riflettendo la luce ricevuta verso S_p . La luce riflessa da un oggetto è caratterizzata dall'essere riflessa o diffusamente o specularmente. La luce riflessa diffusamente è luce che è penetrata sotto la superficie dell'oggetto, è stata assorbita e

quindi riemessa. La superficie sembra ugualmente illuminata da qualsiasi punto si guardi l'oggetto. La legge del coseno di Lambert viene usata per descrivere la riflessione della luce da una superficie perfettamente diffondente, o superficie lambertiana. L'intensità E della luce riflessa da un diffusore perfetto è data da

$$E = \rho E_O \cos \theta$$

dove E_O è l'intensità della sorgente di luce; ρ è l'albedo che varia da 0 (per superfici perfettamente nere) a 1 (per superfici puramente bianche); θ è l'angolo tra la direzione della luce e la normale alla superficie.
La luce riflessa specularmente è riflessa dalla superficie esterna dell'oggetto. In questo caso l'energia della luce riflessa è concentrata principalmente in una direzione particolare, quella in cui il raggio riflesso è nello stesso piano del raggio incidente e soddisfa la condizione che l'angolo di riflessione è uguale all'angolo di incidenza. Questo è il comportamento di uno specchio perfetto.

II.6 Sistema per la Stereo Visione

Un sistema per la stereo visione è, solitamente, un agente software in grado di prendere in ingresso due immagini e di produrre come output una mappa di disparità per tali immagini. La mappa di disparità può essere pensata come una matrice M di interi di dimensione $W \times H$, dove W ed H sono rispettivamente l'ampiezza e l'altezza dell'immagine.

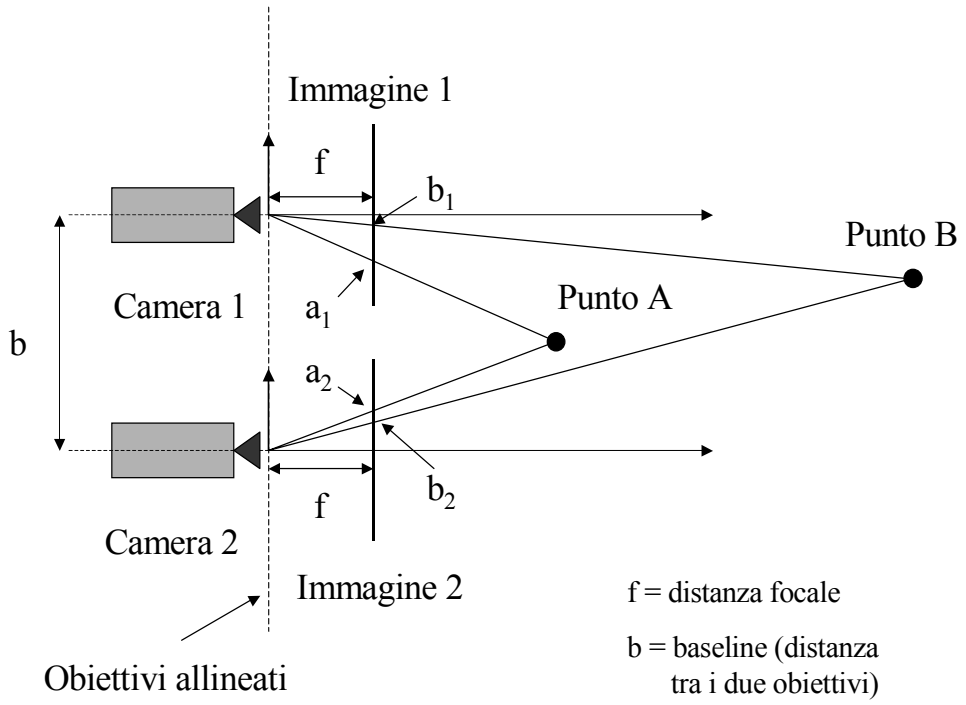


Figura II.8 – Sistema di telecamere con obiettivi allineati

Ogni intero *d* presente nelle celle di *M* rappresenta la distanza (in pixel) tra il pixel di riferimento *p*, preso dalla prima immagine, ed il pixel *p'*, appartenente alla seconda immagine, che più di ogni altro "assomiglia" al pixel *p*.
Più è alto il valore della disparità *d*, più il punto tridimensionale, rappresentato univocamente dalla coppia di pixel *p* e *p',* si trova vicino alle due camere. Questo è vero a condizione che le due camere abbiano i propri obiettivi allineati e lievemente distanti come in figura II.8.
La disparità *d(A)* del punto *A* deve essere maggiore rispetto alla disparità *d(B)* del punto *B*, poiché evidentemente *A* si trova più vicino alle due camere. Definiamo a_1 il punto proiezione di *A* su Immagine 1, a_2 il punto proiezione di *A* su immagine 2, b_1 il punto proiezione di *B* su immagine 1 e b_2 il punto proiezione di *B* su immagine 2.
Si assuma, inoltre, che tutti i punti a_1, a_2, b_1 e b_2 abbiano lo stesso valore di ordinata. Questa assunzione è lecita se si valuta applicabile il vincolo epipolare (visto in precedenza nei paragrafi II.1 e II.4]; in tal caso i quattro punti si trovano sulla stessa linea orizzontale. Calcoliamo i due valori di disparità:

disparità per il punto *A*: $d(A) = a_1 - a_2$

disparità per il punto *B*: $d(B) = b_1 - b_2$

Troviamo $d(A) > d(B)$, poiché $a_1 > b_1$ e $a_2 < b_2$.
E' bene notare che vanno considerati soltanto i valori di ascissa dei quattro punti, e tali valori possono essere confrontati facendo riferimento alla figura II.9.

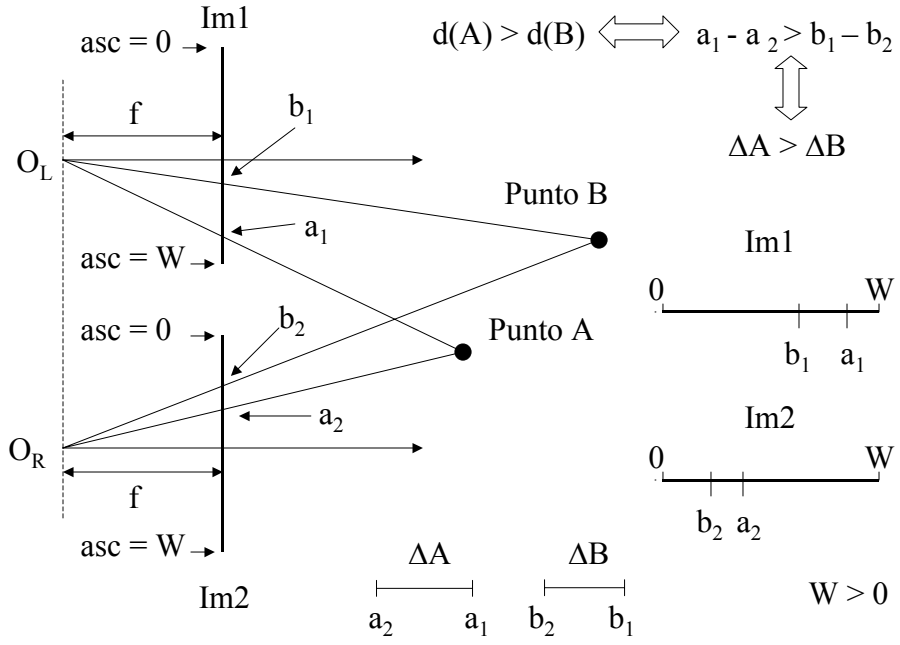


Figura II.9 – Calcolo della disparità

Una volta che la matrice *M* è stata completamente riempita, cioè una volta che per ogni pixel nella immagine di riferimento è stato trovato il corrispettivo nella seconda immagine, abbiamo una mappa della disparità della scena che stiamo analizzando.

Se trasformiamo l'informazione contenuta in *M* sotto forma di numeri interi (che vanno da zero a DMax, con DMax massima disparità ammessa) in una qualche scala di colori, possiamo ottenere una terza immagine che rappresenta gli oggetti della scena con differenti colori, con gradazioni che variano a seconda della distanza di tali oggetti dalle camere. Questa terza immagine verrà, nel seguito, indicata come *immagine di disparità*. Adoperando, ad esempio, una scala di verdi con rapporto crescente tra intensità di verde e valore di disparità, gli oggetti più vicini alle camere appaiono di un verde più acceso, quelli più distanti sono verde più opaco. Una volta prodotta l'immagine di disparità, l'agente software esaurisce il suo compito.

II.7 L'input

L'input per l'agente software è una coppia di immagini. Poiché immagini prodotte da telecamere soffrono inevitabilmente di distorsioni, bisogna che tali immagini originali siano trattate prima di essere date in pasto all'algoritmo.

Figura II.10 – Coppia originale di immagini stereo

Questo trattamento può essere definito *correzione geometrica*. Le distorsioni presenti nelle immagini originali, attraverso un processo di rettificazione (warping), sono corrette e portate in una forma standard.

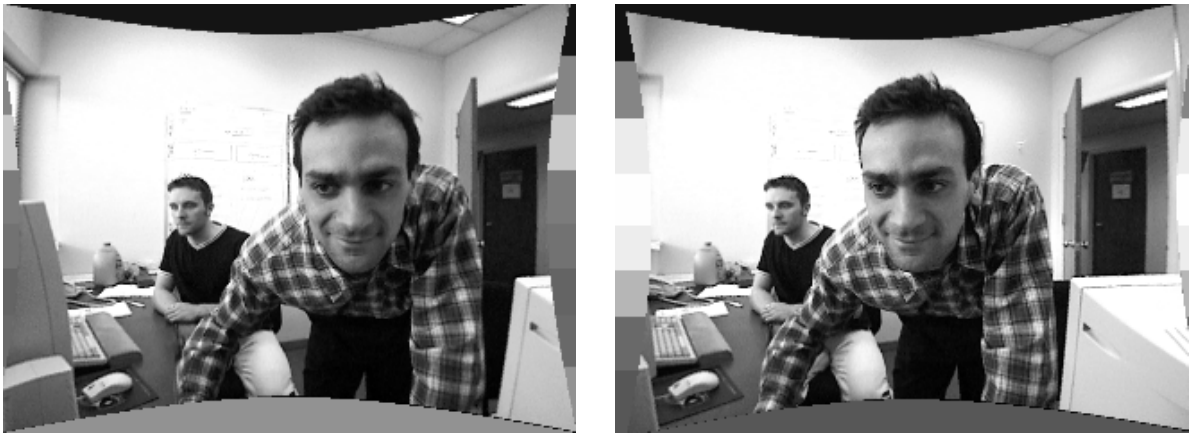


Figura II.11 – Coppia di immagini rettificate

Nel seguito, si darà per scontato che l'input per l'algoritmo di correlazione sia rappresentato da immagini già rettificate.

II.8 L'output

L'output è rappresentato dall'immagine di disparità che scaturisce dal processo di correlazione. Ogni diverso algoritmo di matching produce una differente mappa di disparità e di conseguenza immagini diverse. Si assumerà di rappresentare le informazioni presenti nella mappa con una scala di verdi. Tale scelta è puramente soggettiva: può essere scelta qualsiasi scala di colori, l'importante è che sia possibile distinguere le varie profondità della scena. Viene presentata una serie di immagini di disparità tipo (figure II.12, II.13 e II.14) per le coppie stereo presentate in precedenza, al fine di mostrare ciò che ci attendiamo come output dall'agente.

Immagine sinistra

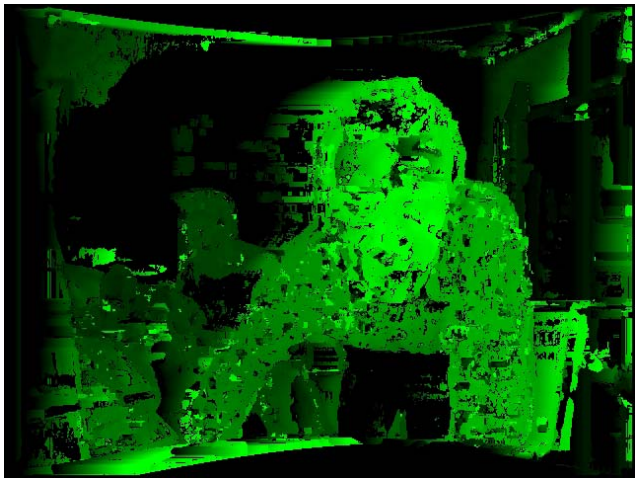


Immagine di disparità

Figura II.12 – L'immagine di disparità è il prodotto dell'algoritmo di correlazione

Immagine sinistra

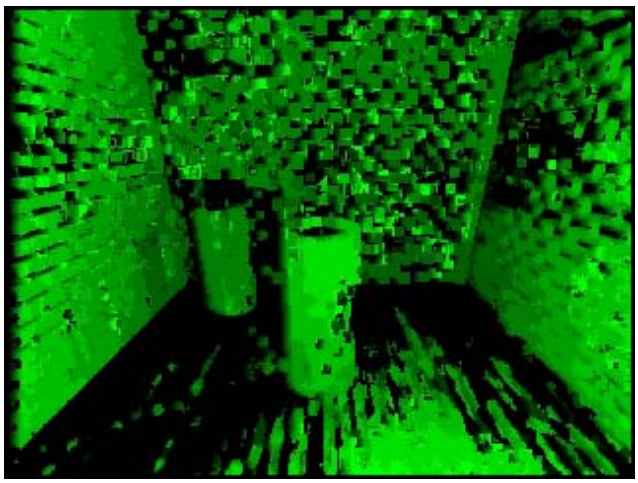


Immagine di disparità

Figura II.13 – Immagine di disparità ottenuta da una coppia virtuale

Immagine sinistra

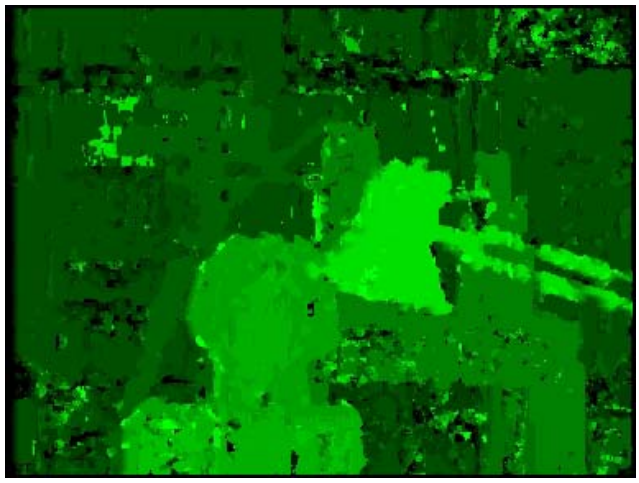


Immagine di disparità

Figura II.14 – Immagine di disparità ottenuta dalla coppia stereo di Tsukuba

II.9 Le tecniche di corrispondenza

La disparità di una coppia coniugata di punti può essere determinata adoperando diversi metodi, sfruttando numerosi vincoli. Tutti questi metodi cercano di accoppiare i pixels dell'immagine di riferimento con i corrispondenti pixels nell'altra immagine. I vincoli, così come i vari metodi, possono essere suddivisi, per semplicità, in *locali* e *globali*.

II.9.1 I vincoli locali

I vincoli locali possono essere applicati su un piccolo numero di pixels che circondano il pixel di interesse. Ad esempio, se i pixels che sono immediatamente adiacenti il pixel *p* di interesse hanno tutti disparità *d*, molto probabilmente anche *p* avrà disparità *d*. Se esprimiamo questo concetto attraverso un vincolo, chiamiamolo *vincolo di continuità a tratti della disparità*, stiamo adottando un vincolo locale.

Mappa di disparità

i - 1	20	20	20
i	20	?	20
i + 1	20	20	20
	j - 1	j	j + 1

p ha coordinate (i, j)

Molto probabilmente la disparità di *p* sarà 20

Figura II.15 – Continuità della disparità per un punto dell'immagine

Il vincolo è locale poiché dipende dal valore di disparità che può essere calcolato in un intorno di p. Se cambiassimo punto di interesse, ad esempio p', quel che era valido per p non sarà più valido per il nuovo punto di interesse p': ecco il motivo della denominazione "locale".

II.9.2 I vincoli globali

I vincoli globali si riferiscono alla direzione di ricerca dei punti coniugati (scan line) o all'immagine nella sua interezza. Il più noto ed usato dei vincoli sulla direzione di ricerca è il vincolo epipolare. Un vincolo che può essere applicato all'immagine intera è il vincolo di luminosità costante: la luminosità di un punto è la stessa nelle due immagini che lo rappresentano. L'adozione del già citato modello Lambertiano (si veda il paragrafo II.5) è un ulteriore esempio di vincolo globale.

II.10 I metodi locali

I metodi locali di correlazione possono essere molto efficienti, tuttavia essi sono notevolmente sensibili alla presenza di regioni delle immagini la cui disparità è ambigua (regioni con presenza di occlusioni o regioni con texture uniforme). E' possibile suddividere questi metodi in tre ampie classi: *block matching*, *gradient methods* e *feature matching*.

II.10.1 Il block matching

I metodi di block matching, detti anche metodi area-based, cercano di stimare la disparità di un punto p nella prima immagine comparando una piccola regione presa attorno a p (area di riferimento o finestra fissa) con una serie di altre regioni, delle stesse dimensioni dell'area di riferimento, estratte dalla seconda immagine (frames di ricerca). Il valore per la funzione di corrispondenza è ottenuto mettendo a confronto la finestra fissa della prima immagine con una finestra scorrevole nella seconda immagine. La finestra scorrevole viene spostata sulla seconda immagine, attraverso incrementi interi, lungo la linea epipolare di p: ogni singolo incremento genera un nuovo frame di ricerca. In base al vincolo epipolare, i frames di ricerca avranno per centro un punto con coordinate di ascissa variabile, ma con ordinata fissa ed uguale al valore di ordinata per p. Il vincolo epipolare, poiché fissa il valore di ordinata, riduce la complessità della ricerca da bidimensionale a mono-dimensionale.
Una volta trovato il frame di ricerca che più assomiglia alla finestra di riferimento, il baricentro p' di tale frame sarà il pixel corrispondente a p. E' fondamentale stabilire in che modo si intende attribuire un grado di somiglianza per ogni singolo frame che si processa. In altre parole, deve essere definita una qualche *misura di similarità*, in modo che il confronto tra la finestra di riferimento ed i frames possa essere legato ad una scala di valori con cui produrre la curva di corrispondenza. Così facendo, il confronto tra la finestra di riferimento ed i vari frames genererà una curva dei valori

di corrispondenza (funzione di corrispondenza) per ogni punto della seconda immagine che risiede sulla linea epipolare di *p*. La disparità corretta può essere definita come la differenza (in valore assoluto) tra il valore di ascissa di *p* ed il valore di ascissa del centro *p′* del frame di ricerca in relazione al quale la funzione di corrispondenza presenta il picco più alto. Alcune possibili misure di similarità sono la *correlazione* e le metriche legate al rango (*rank metrics*).

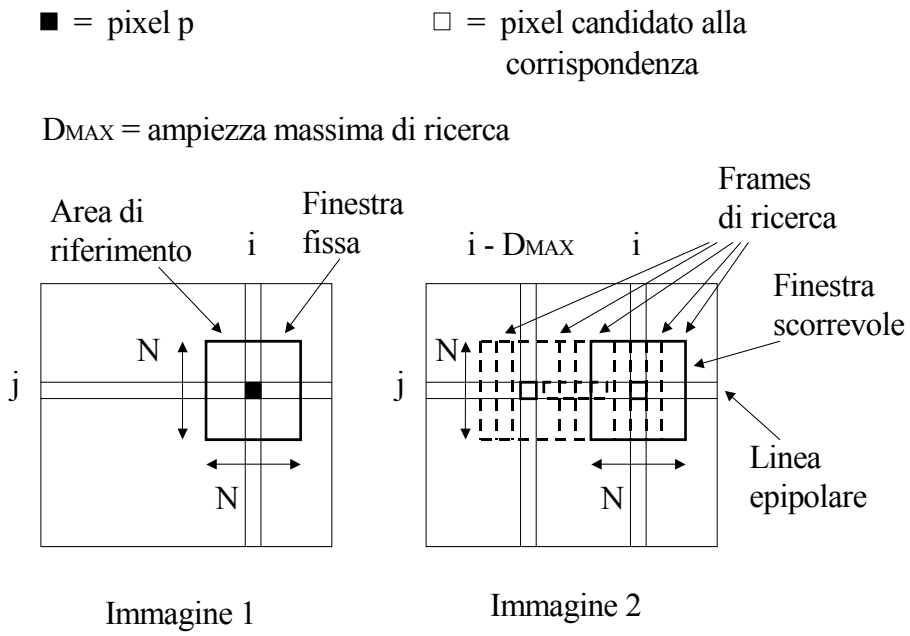


Figura II.16 – Il block matching

II.10.2 La Correlazione

La correlazione è una tecnica di facile implementazione che genera una mappa di disparità densa, ovvero viene trovata la disparità per ogni punto dell'immagine di riferimento; tuttavia, proprio perché processa tutti i pixels, la correlazione risulta relativamente lenta, lentezza che mal si addice a sistemi real-time. Il matching della finestra di riferimento con i vari frames viene effettuato confrontando algebricamente i valori (per immagini in bianco e nero si tratta dei valori GRAYSCALE, per immagini a colori si utilizzano i valori RGB) dei pixels presenti nella finestra e nel frame. Focalizziamo l'attenzione sulla figura II.17 a pagina seguente.
L'immagine sinistra è l'immagine di riferimento, mentre sull'immagine destra devono essere scelti i frames di ricerca con cui confrontare la finestra di riferimento. Se si presta un minimo di attenzione all'immagine inferiore (quella destra) si nota che la scena è traslata verso sinistra in relazione all'immagine di riferimento. La traslazione è dovuta alle posizioni relative degli obiettivi delle due telecamere: l'obiettivo più a destra necessariamente vedrà gli oggetti leggermente shiftati rispetto all'obiettivo sinistro. Questo vuol dire che un punto *p* di coordinate *(x, y)* preso sull'immagine sinistra avrà un coniugato sull'immagine destra in posizione *(x – d, y)*, con *d* valore di disparità che varia tra 0 e DMax.

La ricerca andrà fatta all'indietro, nel senso che i frames di ricerca avranno di volta in volta un centro con valore di ascissa decrescente.

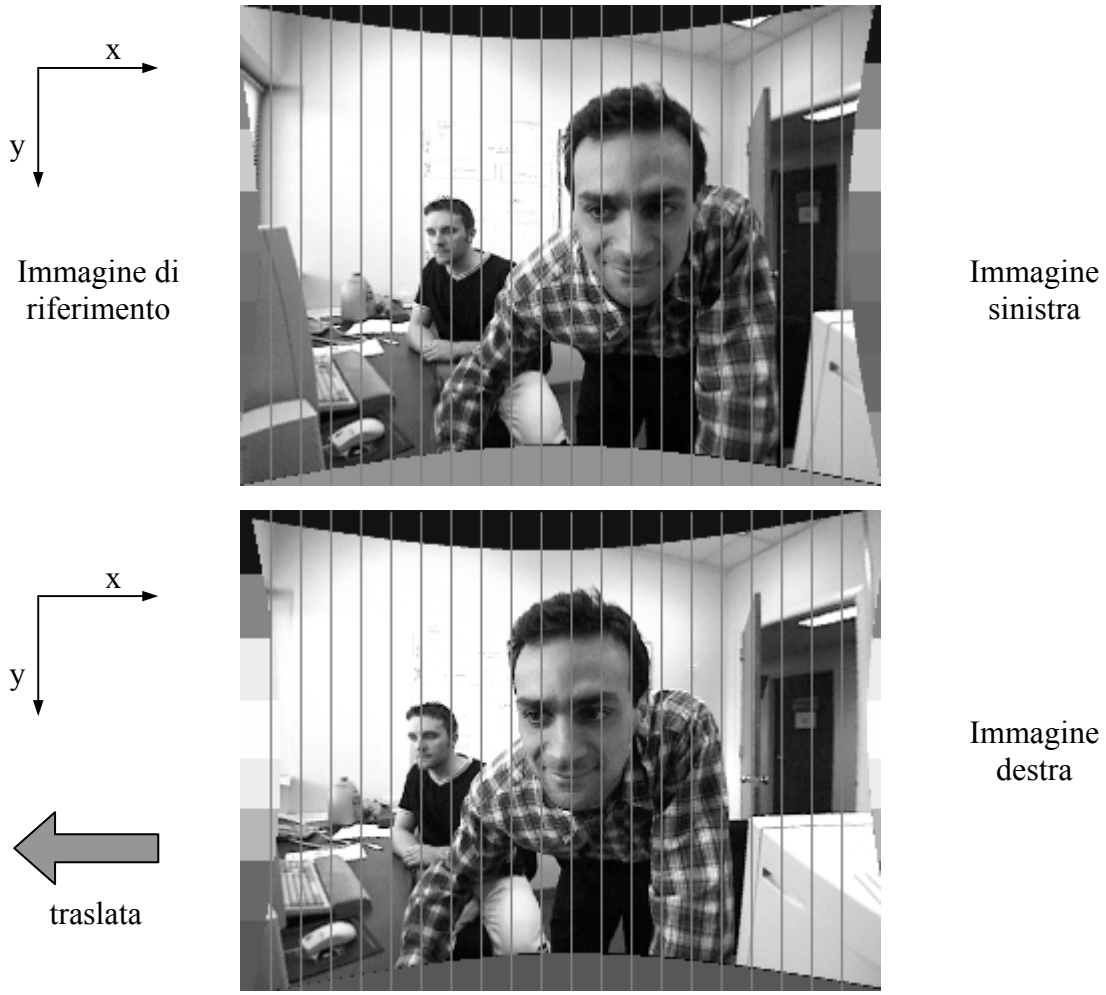


Figura II.17 – Calcolo della disparità per una coppia stereo

Se si scegliesse di prendere come immagine di riferimento quella destra, il coniugato del generico punto *p* (questa volta *p* sarà sull'immagine destra) avrà coordinate diverse, precisamente *(x + d, y)*. Una volta individuata la rosa dei possibili candidati ad essere il frame di ricerca che più assomiglia alla finestra di riferimento, bisogna trovare in questo insieme il miglior match possibile. Esistono diversi metodi per farlo, ne vengono presentati alcuni.

SSD Sum of Squared Difference

$$C_{SSD}(x, y, d) = min \{ \sum_{i,j} [I_1(x + i, y + j) - I_2(x - d + i, y + j)]^2 \} \qquad (2.2)$$

SAD Sum of Absolute Difference

$$C_{SAD}(x, y, d) = min \{ \sum_{i,j} | I_1(x + i, y + j) - I_2(x - d + i, y + j) | \} \qquad (2.3)$$

NCC Normalized Cross Correlation (2.4)

$$C_{NCC}(x, y, d) =$$

$$\max \left\{ \frac{\sum_{i,j} [I_1(x+i, y+j) - \bar{I}_1(x, y)] \cdot [I_2(x-d+i, y+j) - \bar{I}_2(x+d, y)]}{\left\{ \sum_{i,j} [I_1(x+i, y+j) - \bar{I}_1(x, y)]^2 \cdot \sum_{i,j} [I_2(x-d+i, y+j) - \bar{I}_2(x+d, y)]^2 \right\}^{1/2}} \right\}$$

L'indice i varia tra $-\lfloor N/2 \rfloor$ e $+\lfloor N/2 \rfloor$ con N ampiezza della finestra, l'indice j varia tra $-\lfloor M/2 \rfloor$ e $+\lfloor M/2 \rfloor$ con M altezza della finestra.
$I_1(x, y)$ e $I_2(x, y)$ denotano i valori di intensità del pixel *(x, y)* per la prima (I_1) e la seconda (I_2) immagine, d è il valore di disparità.
$\bar{I}_k(x, y)$ con $k = 1, 2$ è la media, all'interno della finestra di dimensione $N \times M$, dei valori di luminosità dei pixels dell'immagine k.
Procediamo con l'analisi di un esempio che sfrutta il metodo SSD, cioè l'equazione (2.2) presentata sopra.

Esempio II.1 *Si considerino due ipotetiche matrici che devono essere correlate (figura II.18). Ogni cella (i, j) di una matrice contiene un intero che rappresenta il valore di colore per il punto (i, j) dell'immagine a cui si riferisce la matrice.*
Si supponga di voler trovare per il punto p, di coordinate riga-colonna (4, 5), il proprio coniugato p'. Inoltre, si suppongano le dimensioni della finestra di correlazione pari a 3 x 3 e la massima ampiezza di ricerca (DMax) pari a 2.

Matrice 1
immagine sinistra

	0	1	2	3	4	5	6	7	8	9	10	11
0	1	3	7	6	4	8	9	6	5	4	6	7
1	5	8	5	3	5	7	9	8	6	4	3	4
2	6	7	9	0	6	6	3	5	6	7	8	9
3	7	5	4	3	4	4	7	6	5	4	4	9
4	7	3	7	8	9	9	1	2	3	0	0	0
5	4	5	6	7	8	4	5	6	7	7	5	1
6	9	4	6	3	4	6	7	8	3	4	6	7
7	2	3	4	5	6	7	7	5	4	3	2	4
8	3	7	4	9	9	9	9	5	5	5	3	4
9	5	6	7	5	4	6	8	9	4	5	7	7
10	8	5	8	5	8	5	6	5	7	8	5	8
11	5	5	3	4	8	8	9	5	3	0	0	0

Matrice 2
immagine destra

	0	1	2	3	4	5	6	7	8	9	10	11
0	1	4	4	7	9	0	0	8	5	7	3	2
1	2	6	5	3	5	9	8	8	8	8	5	4
2	5	6	8	9	4	5	2	3	4	1	1	1
3	4	5	6	7	8	9	0	8	5	4	4	1
4	1	3	1	1	9	6	9	2	3	5	0	4
5	4	5	5	7	8	9	0	8	5	3	6	7
6	3	2	6	3	4	3	2	4	5	6	4	3
7	4	5	6	5	7	7	5	6	4	7	2	5
8	5	7	6	6	5	5	6	7	9	6	3	2
9	3	4	5	6	4	4	8	4	6	7	8	7
10	2	4	8	4	4	4	4	5	2	3	4	6
11	5	5	6	4	6	8	6	5	6	0	6	0

Figura II.18 – Le due matrici da correlare

La figura II.19 (a pagina seguente) mostra i passi da affrontare per trovare p' a partire da p.

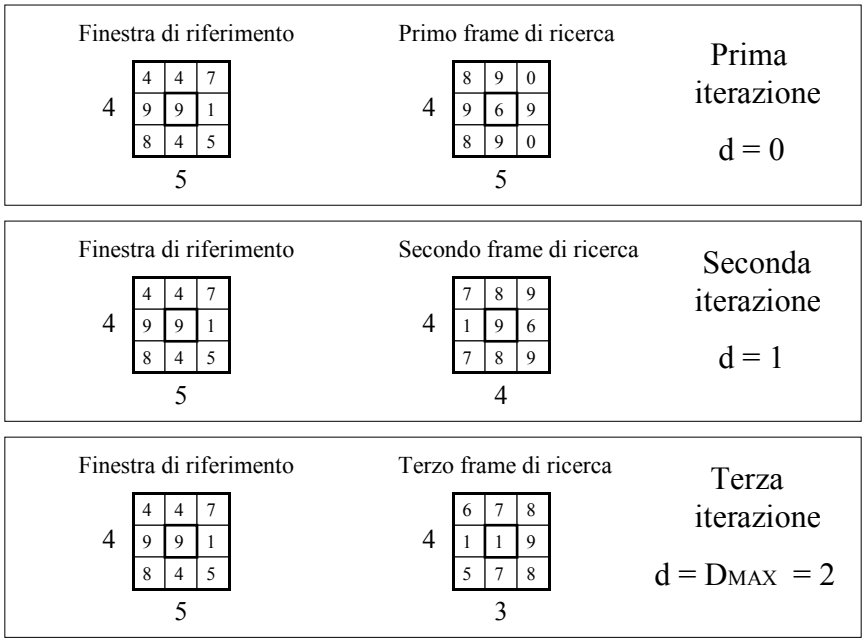


Figura II.19 – I passi della correlazione per il punto p ≡ (4, 5)

Il vincolo epipolare garantisce che p' abbia come ordinata il valore 4, mentre per il valore di ascissa devono essere prese in considerazione diverse ipotesi. Poiché il valore fissato per l'ampiezza di ricerca (DMax) è 2, la finestra di riferimento dovrà essere confrontata con tre frames di ricerca.

Prima iterazione: $S_1 = (4-8)^2 + (4-9)^2 + (7-0)^2 + (9-9)^2 + (9-6)^2 + (1-9)^2 + (8-8)^2 + (4-9)^2 + (5-0)^2 = 213$ $d = 0$

Seconda iterazione: $S_2 = (4-7)^2 + (4-8)^2 + (7-9)^2 + (9-1)^2 + (9-9)^2 + (1-6)^2 + (8-7)^2 + (4-8)^2 + (5-9)^2 = 151$ $d = 1$

Terza iterazione: $S_3 = (4-6)^2 + (4-7)^2 + (7-8)^2 + (9-1)^2 + (9-1)^2 + (1-9)^2 + (8-5)^2 + (4-7)^2 + (5-8)^2 = 233$ $d = DMAX = 2$

La disparità per p è 1. Infatti, adoperando il metodo SSD, un valore di somma nullo indica la corrispondenza perfetta, il miglior match è quello con il valore di somma più basso.
Vediamo cosa accade utilizzando, al posto dell'SSD, il SAD. Quest'ultimo metodo è da preferirsi se si intende risparmiare risorse per la computazione o se le risorse sono, per qualche motivo, limitate. Infatti, il calcolo dei quadrati nell'SSD è certamente più oneroso sia in termini di risorse adoperate, sia in termini di tempo di elaborazione rispetto al semplice valore assoluto che adopera il SAD.

Prima iterazione: $S_1 = |4-8| + |4-9| + |7-0| + |9-9| + |9-6| + |1-9| + |8-8| + |4-9| + |5-0| = 37$ $d = 0$

Seconda iterazione: $S_2 = |4-7| + |4-8| + |7-9| + |9-1| + |9-9| + |1-6| + |8-7| + |4-8| + |5-9| = 31$ $d = 1$

Terza iterazione: $S_3 = |4-6| + |4-7| + |7-8| + |9-1| + |9-1| + |1-9| + |8-5| + |4-7| + |5-8| = 39$ $d = DMAX = 2$

Anche adoperando il SAD il secondo frames risulta essere il più somigliante alla finestra di riferimento; si può concludere affermando che il coniugato di $p \equiv (4, 5)$ *è* $p' \equiv (4, 4)$. □

L'esempio II.1 sopra riportato, volutamente molto semplice, potrebbe portare a considerare la correlazione un problema di facile soluzione. In realtà, esistono tutta una serie di considerazioni che è necessario tenere ben presenti.

1. È noto che la probabilità di un match sbagliato decresce in proporzione all'aumentare delle dimensioni della finestra di correlazione [17].

2. L'uso di finestre di grosse dimensioni permette di incrementare l'attendibilità dei risultati, poiché è possibile confrontare aree più grandi che contengono maggiori informazioni sulla localizzazione del pixel di interesse e ridurre, in tal modo, l'effetto del rumore che sempre accompagna una immagine. Tuttavia, l'impiego di grandi finestre porta ad una perdita di accuratezza e alla possibile omissione di importanti caratteristiche dell'immagine.

3. Finestre di piccole dimensioni aumentano la risoluzione dell'immagine di disparità, poiché sono in grado di gestire anche piccoli oggetti presenti nella scena, che altrimenti sarebbero trascurati da finestre più ampie. Per finestre più piccole, però, aumentano le probabilità di falsi accoppiamenti ed è necessaria l'introduzione di un valore di soglia per correggere la correlazione. Tale valore è arbitrario e, di conseguenza, di difficile individuazione.

4. La correlazione assume che i punti all'interno della finestra di riferimento e nei frames di ricerca abbiano profondità costante; ciò non è vero, invece, nel caso in cui la finestra o un frame coprano una regione dell'immagine in cui siano presenti due oggetti a diversa distanza dagli obiettivi delle telecamere.

Osservazione II.1 *In generale, variazioni repentine di profondità introducono errori nel calcolo della disparità con la correlazione*.

Una particolare situazione si ha se una finestra coincide con una regione che presenta forti differenze di profondità: in questo caso, una parte della finestra influenzerà negativamente il risultato.

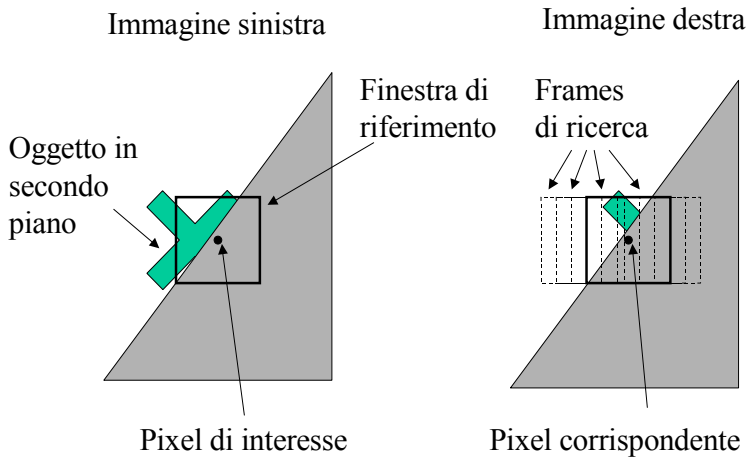


Figura II.20 – La correlazione sul bordo di un oggetto

La figura II.20 mostra l'eventualità in cui la regione sinistra della finestra di riferimento contenga parte di un oggetto in secondo piano (bassa disparità) e parte di un oggetto in primo piano (alta disparità); inoltre, l'oggetto in secondo piano risulta parzialmente occluso per i frames di ricerca. Conseguentemente, la parte sinistra della finestra di riferimento introduce un errore nei calcoli della disparità. Per di più, si può notare che la dimensione della parte occlusa dipende dalla differenza di disparità e dalle dimensioni della finestra di riferimento. L'uso di una finestra più piccola riduce il problema, poiché una finestra di dimensioni minori riduce la probabilità di contenere al suo interno la discontinuità di profondità.

II.10.3 Le metriche legate al rango

In alternativa alla correlazione, sono stati proposti metodi per il calcolo della corrispondenza che sfruttano l'applicazione alle immagini, in una fase antecedente il matching, di trasformazioni locali non parametriche.
Nel tentativo di eliminare la sensibilità al rumore, una trasformata legata al rango (rank transform) è localmente applicata alle regioni di entrambe le immagini.
La rank transform per una regione R che circonda un pixel p è definita come il numero di pixels in R per i quali l'intensità è minore di quella di p. I valori risultanti sono basati sull'ordine relativo dei pixels in R anziché sul valore assoluto delle intensità. L'intervallo di valori che può assumere la trasformata è limitato dalla dimensione della finestra di correlazione, per cui si otterrà un insieme di valori molto

livellato che è di aiuto nell'eliminare gli errori dovuti alle occlusioni. Infatti, per quanto grande, la finestra di ricerca è di ampiezza limitata.

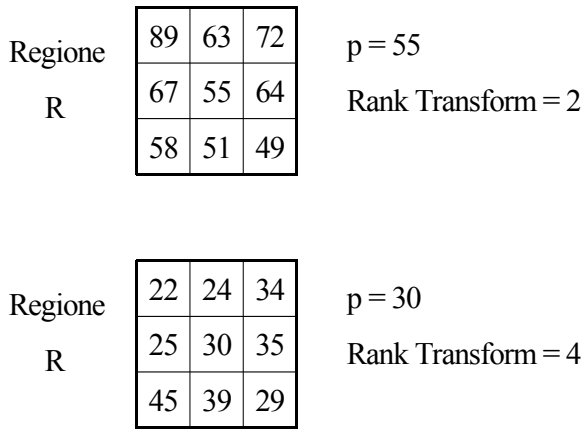


Figura II.21 – Esempi di rank transform

Dopo aver applicato la rank transform, si procede adoperando la correlazione, tipicamente facendo uso della SAD.

Rank Transform (2.5)

$$\sum_{u,v} [\, I'_1(u, v) - I'_2(u - d, v)\,]$$

$$I'_k(u, v) = \sum_{m,n} I_k(m, n) < I_k(u, v)$$

Se è vero che l'uso della trasformata legata al rango riduce la sensibilità ai rumori, è anche vero che essa riduce un po' la potenza computazionale dell'algoritmo di corrispondenza poiché fa perdere parte dell'informazione. L'ordine relativo di tutti i pixels che giacciono intorno al pixel *p* a cui si applica la trasformata è codificato tramite un unico valore, per questo è possibile conoscere il numero dei pixels che hanno intensità minore di *p* ma non la loro posizione all'interno della regione. Per limitare la perdita d'informazione, è stata proposta una trasformata che possa fornire dati anche sulla posizione relativa dei pixels nella regione di interesse, una sorta di censimento di tali pixels. La trasformata che effettua questo censimento (detta proprio *census transform*) conserva l'informazione sulla distribuzione spaziale dei pixels producendo una stringa di numeri binari. La lunghezza *L* della stringa dipende dall' area *A* della finestra di correlazione ed è pari ad $L = A - 1$ poiché la posizione del pixel di interesse non deve essere codificata. La procedura di corrispondenza prosegue con il calcolo della distanza di Hamming tra le due stringhe di bit.
E' evidente che, aumentando l'informazione da memorizzare, la census transform risulti molto più costosa, in termini di risorse e tempo di esecuzione, rispetto alla rank transform di un fattore proporzionale alla dimensione della finestra di ricerca.

Regione R

89	63	72
67	55	64
58	51	49

p = 55

Census Transform = 0000011

Regione R

22	24	34
25	30	35
45	39	29

p = 30

Census Transform = 11010001

Figura II.22 – Esempi di census transform

Census Transform (2.6)

$$\sum_{u,v} HAMMING\,[\, I'_1(u, v)\,,\, I'_2(u - d, v)\,]$$

$$I'_k(u, v) = BITSTRING_{m,n}\,[\, I_k(m, n) < I_k(u, v)\,]$$

Alcuni studi indicano che i metodi basati sulle trasformate permettono di ridurre il rumore dovuto alle distorsioni radiometriche ed alle occlusioni [17].

II.10.4 Gradient methods

I metodi basati sul calcolo di un gradiente cercano di determinare le disparità tra due immagini formulando una equazione differenziale che collega la disparità alla variazione della luminosità dell'immagine. La luminosità di un punto della scena viene assunta essere costante in tutte e due le immagini, in questo modo si può calcolare la traslazione orizzontale (disparità) di un punto da una immagine all'altra risolvendo l'equazione differenziale

$$(\nabla_x E)\, v + E_t = 0 \qquad (2.7)$$

dove $\nabla_x E$ denota la componente orizzontale del gradiente dell'immagine, E_t la derivata temporale (qui si riferisce alla differenza di intensità tra l'immagine sinistra e destra) e v la traslazione tra le due immagini.

II.10.5 Feature matching

Il block matching ed i gradient methods sono molto sensibili alle discontinuità di profondità, giacché una finestra di ricerca può contenere punti con diversi valori di profondità. Questi metodi, inoltre, sono molto sensibili alle regioni delle immagini in cui la texture sia molto uniforme.

I metodi feature-based cercano di oltrepassare questi problemi limitando il calcolo della corrispondenza solo ad alcune zone della scena, in cui il calcolo della disparità sia molto attendibile. Queste zone (*features*) sono gli angoli, gli spigoli ed i bordi degli oggetti presenti nella scena. Certamente, questo modo di procedere non permette di realizzare mappe di disparità dense, perché solo una parte dei pixels delle due immagini è coinvolta nel processo di individuazione delle disparità; d'altro canto, il feature matching è estremamente efficiente sotto l'aspetto del tempo di elaborazione.
Negli anni '80 i sistemi di stereo visione basati sul feature matching ricevettero grande attenzione, soprattutto in virtù della loro grande efficienza. Nell'ultimo decennio, invece, sia perché una gran quantità di applicazioni necessita di mappe dense, sia perché i metodi basati sul block matching hanno incrementato la loro efficienza e robustezza, l'interesse per i metodi feature-based è andato scemando. Un tipico approccio per l'implementazione del feature matching è il metodo gerarchico. Questo metodo utilizza quattro tipi di features: linee, vertici, spigoli e superfici. L'algoritmo gerarchico parte analizzando le superfici (livello più alto), fino ad arrivare alle linee (livello più basso).

II.11 I metodi globali

I metodi globali sono molto meno sensibili rispetto ai metodi locali alle regioni che presentano disparità ambigua (occlusioni e regioni a texture uniforme) poiché i vincoli globali che essi sfruttano garantiscono un supporto addizionale per la gestione delle regioni difficilmente trattabili con metodi locali. L'uso di questi vincoli globali, però, fa diventare la complessità computazionale dei metodi globali significativamente più elevata rispetto a quella dei metodi locali. Per tale motivo, sono stati proposti diversi metodi matematici per ridurre la complessità, sfruttando la decomposizione dei problemi di ottimizzazione in sottoproblemi più piccoli e più semplici da risolvere. La trattazione dei metodi globali non rientra tra gli scopi di questo scritto, quindi essi non verranno più presi in considerazione.

II.12 Complessità computazionali

- Il block matching ha un costo computazionale di $O(NDn)$ dove N è il numero totale dei pixels che compongono l'immagine, D è il massimo valore di disparità ammesso (altrimenti detto DMax) ed n è l'area della finestra di ricerca. Questa complessità è evidentemente molto elevata; per fortuna bisogna aver presente che essa viene fuori solo se si compie una ingenua implementazione della correlazione. Infatti, la cieca applicazione del block matching genera un elevato numero di calcoli ridondanti. Queste ridondanze possono essere eliminate gestendo con attenzione i calcoli per ogni blocco che viene processato: adoperando piccoli accorgimenti, è possibile ridurre la complessità fino a $O(ND)$

rendendo il calcolo delle corrispondenze indipen-dente dalla dimensione della finestra di ricerca [17].

- I gradient methods hanno una complessità di solo $O(N)$, tuttavia soltanto le traslazioni nella direzione del gradiente possono essere stimate con attendibilità. E' necessario inserire un vincolo addizionale che possa aumentare l'accuratezza dei calcoli, tale inserimento porta la complessità computazionale a $O(Nn)$, come per il block matching.

- Il feature matching ha una complessità dipendente dal numero di caratteristiche dell'immagine (features) che vengono prese in considerazione: valutando il metodo gerarchico la complessità per ogni livello è $O(N^4)$, con N numero di features esaminate a quel livello [17].

Capitolo III

Stato dell'arte delle tecniche steganografiche

Capitolo III

Stato dell'arte delle tecniche steganografiche

Questo capitolo offre una panoramica sulle odierne tecniche steganografiche a nostra disposizione. Il materiale seguente è fortemente ispirato a diversi articoli [4, 6, 21], i quali costituiscono una guida completa dei metodi disponibili oggi e delle tecniche per attaccarli con successo.

III.1 I file immagine

Per un computer un'immagine è un array di numeri che rappresentano l'intensità della luce in vari punti (*pixels*). Questi pixels formano il *raster data* dell'immagine. Un'immagine comune ha dimensioni pari a 640 × 480 pixels e 256 colori (o 8 bit per pixel). Una siffatta immagine potrebbe "pesare" circa 300 kilobit.
Le immagini digitali sono tipicamente salvate in formati da 24 bit o da 8 bit per pixel. È naturale che, rispetto ad una immagine ad 8, un'immagine a 24 bit fornisca uno spazio molto maggiore per nascondere l'informazione: tale spazio può essere abbastanza ampio (ad eccezione delle immagini JPEG) per inserire un testo molto lungo. In generale, tutte le variazioni di colore per i pixel sono derivate dai tre colori primari: rosso (*red*), verde (*green*) e blu (*blue*). Ogni colore primario è rappresentato da 1 byte: le immagini a 24 bit usano 3 bytes per pixel per rappresentare un valore del colore. Questi 3 bytes possono essere rappresentati in valori esadecimali, decimali e binari (si veda l'esempio III.1).

Esempio III.1 *In molte pagine web, il colore dello sfondo è rappresentato da un numero a sei cifre esadecimali - in realtà 3 coppie che rappresentano rosso, verde e blu. Uno sfondo bianco avrebbe il valore FFFFFF: 100% rosso (FF), 100% verde (FF) e 100% blu (FF). il suo valore decimale è 255, 255, 255 ed il suo valore binario è 11111111, 11111111, 11111111, che sono appunto i tre bytes che compongono il bianco. Questa definizione di sfondo bianco è analoga alla definizione di colore di un singolo pixel in una immagine.* □

La rappresentazione dei pixel contribuisce alla dimensione del file. Ad esempio, si supponga di avere una immagine a 24 bit, 1024 × 768 pixel – una risoluzione comune per le grafiche ad alta risoluzione. Una immagine del genere ha più di due milioni di pixels, ognuno dei quali con una definizione tale da produrre un file di oltre 2 Mbytes. Poiché queste immagini a 24 bit sono ancora relativamente poco comuni su Internet, la loro dimensione può essere un parametro importante durante la trasmis-

sione. Una qualche forma di compressione del file, sarebbe indubbiamente vantaggiosa, per non dire necessaria, al fine di trasmettere un simile file.

III.2 La compressione

Esistono due tipi di compressione: *lossless* (che significa senza perdita) e *lossy* (con perdita). Entrambi i metodi consentono un risparmio di memoria ma con risultati diversi, che interferiscono con la informazione nascosta, quando tale informazione non sia compressa.
La compressione lossless permette di ricostruire **esattamente** il messaggio originale, perciò si preferisce quando l'informazione originale deve rimanere intatta (come nel caso delle immagini steganografiche). Tale compressione è tipica delle immagini salvate come GIF (Graphic Interchange Format) e BMP a 8 bit.

Esempio III.2 *Il formato grafico GIF è molto semplice ed è costituito da due elementi:*

1. *una "tavolozza" di colori (in inglese palette, si veda il paragrafo III.3.1 per maggiori dettagli) che contiene fino a 256 distinti colori, sull'insieme dei 2^{24} possibili;*
2. *una matrice di puntatori alla tavolozza, compressa con l'arcinoto algoritmo Lempel – Ziv – Welch (LZW), che è un algoritmo senza perdita di dati.*

Ogni pixel dell'immagine è rappresentato da un puntatore ad uno dei 256 colori della tavolozza. □

La compressione lossy, invece, occupa meno spazio ma non mantiene l'integrità dell'immagine originale. Questo metodo è impiegato tipicamente con immagini di tipo JPEG (per i dettagli della compressione JPEG si veda il paragrafo IV.9.1). Grazie agli algoritmi di compressione lossy, il formato JPEG consente di ottenere approssimazioni molto vicine a fotografie digitali di alta qualità, ma non un duplicato esatto. Da qui il termine compressone lossy.

III.3 L' inserimento dei dati

Il processo di inserimento dei dati (*embedding data*), che corrisponde al nascondere i dati in una immagine, richiede due file. Il primo è l'immagine non sospetta che conterrà l'informazione nascosta, chiamata cover image; il secondo è il messaggio segreto, ovvero l'informazione da nascondere. Un messaggio può essere un plaintext, un ciphertext, un'altra immagine o qualsiasi altra cosa che possa essere contenuta in un flusso di bit. Una volta combinati, la cover image ed il messaggio in essa inserito formano una stego image. Una *stego key* (una sorta di password) può, in aggiunta, essere usata per nascondere, e poi successivamente decodificare, il messaggio.

Per molti software steganografici non si supporta né si consiglia l'uso di immagini JPEG, ma si consiglia invece l'uso di immagini lossless a 24 bit, come ad esempio immagini BMP: si tratta di una pesante limitazione, che verrà superata in questa tesi.
Un'altra buona alternativa alle immagini a 24 bit è costituita dalle immagini a 256 colori o grayscale.

III.3.1 Steganografia su palette

La palette (tavolozza) è un sottoinsieme prestabilito di colori. Nei formati che ne fanno uso (abbiamo visto GIF), i pixel della bitmap sono vincolati ad assumere come valore uno dei colori presenti nella palette: in questo modo è possibile rappresentare i pixel con dei puntatori alla palette, invece che con la terna esplicita RGB (red, green e blue). Ciò in genere permette di ottenere dimensioni inferiori della bitmap, ma il reale vantaggio è dato dal fatto che le schede grafiche di alcuni anni fa utilizzavano proprio questa tecnica e quindi non potevano visualizzare direttamente immagini con un numero arbitrario di colori. Il caso più tipico è quello delle immagini in formato GIF con palette di 256 colori, ma le palette possono avere anche altre dimensioni. Come è facile immaginare, un'immagine appena prodotta da uno scanner a colori sarà tipicamente costituita da più di 256 colori diversi, tuttavia esistono algoritmi capaci di ridurre il numero dei colori utilizzati mantenendo il degrado della qualità entro limiti accettabili. Si può osservare che, allo stesso modo in cui avviene con il formato JPEG, non è possibile iniettare informazioni sui pixel prima di convertire l'immagine in formato GIF, perché durante il processo di conversione c'è perdita di informazione (osserviamo, anche, che questo non vale per le immagini a livelli di grigi: tali immagini infatti sono particolarmente adatte per usi steganografici). La soluzione che viene di solito adottata per usare immagini GIF come contenitori è dunque la seguente: si riduce il numero dei colori utilizzati dall'immagine a un valore inferiore a 256 ma ancora sufficiente a mantenere una certa qualità dell'immagine, dopodichè si finisce di riempire la palette con colori molto simili a quelli rimasti. A questo punto, per ogni pixel dell'immagine, la palette contiene più di un colore che lo possa rappresentare (uno è il colore originale, gli altri sono quelli simili ad esso che sono stati aggiunti in seguito), quindi abbiamo una possibilità di scelta. Tutte le volte che esiste una possibilità di scelta fra più alternative, è possibile nascondere un'informazione: questo è uno dei principi fondamentali della steganografia. Se le alternative sono *due* possiamo nascondere *un bit* (se il bit è 0, scegliamo la prima, se è 1 la seconda); se le alternative sono *quattro* possiamo nascondere *due bit* (00 → la prima, 01 → la seconda, 10 → la terza, 11 → la quarta) e così via.
La soluzione appena discussa dell'utilizzo di GIF come contenitori è molto ingegnosa ma purtroppo presenta un problema: è facile scrivere un programma che, presa una GIF in ingresso, analizzi i colori utilizzati e scopra le relazioni che esistono tra di essi; se il programma scopre che l'insieme dei colori utilizzati può essere ripartito in sottoinsiemi (cluster) di colori simili, è molto probabile che la GIF contenga informazione steganografata. Di fatto, questo semplice metodo di attacco (che potremmo definire, giusto per intenderci, *clusterizzazione*) è stato portato avanti con

pieno successo in [28], tanto da rendere poco sicuro, in generale, l'uso di immagini GIF come cover medium.
Per mostrare quanto sia ampia la gamma di tecniche steganografiche, accenniamo a un'altra possibilità di nascondere informazioni dentro immagini GIF. Come abbiamo detto, in questo formato viene prima memorizzata una palette e poi la matrice bitmap (compressa con l'algoritmo LZW) dei puntatori alla tavolozza. Se scambiamo l'ordine di due colori della palette e contemporaneamente tutti i puntatori ad essi, otteniamo un file diverso che corrisponde però alla stessa immagine: dal punto di vista dell'immagine, il contenuto informativo dei due file è identico. La rappresentazione di immagini con palette è quindi intrinsecamente ridondante, dato che ci permette di scegliere un qualsiasi ordine dei colori della palette (purché si riordinino correttamente i puntatori ad essi). Se i colori sono 256, esistono 256! modi diversi di scrivere la palette, quindi esistono 256! file diversi che rappresentano la stessa immagine. Inoltre, è abbastanza facile trovare un metodo per numerare univocamente tutte le permutazioni di ogni data palette (basta, per esempio, considerare l'ordinamento sulle componenti RGB dei colori). Dato che abbiamo 256! possibilità di scelta, è possibile codificare $log(256!) = 1683$ bit, cioè 210 byte. Si noti che questo numero è indipendente dalle dimensioni dell'immagine, in altre parole è possibile iniettare 210 bytes anche su piccole immagini del tipo icone 16 × 16 semplicemente permutando in modo opportuno la palette.

III.3.2 EzStego

Il software EzStego, sviluppato da Romana Machado, inserisce il messaggio segreto in un file GIF, senza alcuna informazione sulla sua lunghezza. EzStego non modifica la tavolozza dei colori, creando, invece, una copia ordinata della palette.

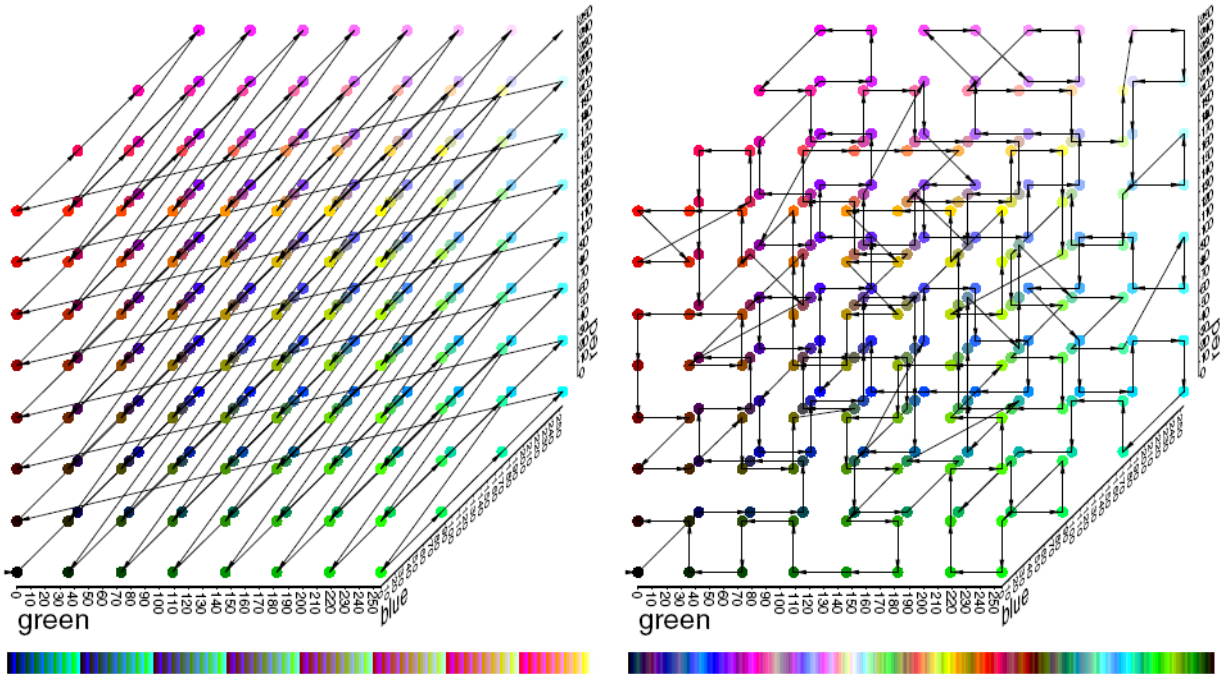


Figura III.1 – Ordine dei colori nella palette (a sinistra) e ordine dei colori fatto da EZStego (a destra), tratto da [19]

Questa copia è costruita in modo che sia minima la differenza visiva tra due colori in essa adiacenti (si veda la barra alla base dei due grafici in figura III.1). Ordinare in base ai soli valori di luminanza è sconsigliabile, perché due colori con la stessa luminanza potrebbero essere radicalmente diversi alla vista. Si può immaginare ogni colore come un punto in uno spazio tridimensionale, il cubo RGB. Nella parte sinistra della figura III.1 i colori appaiono molto più ordinati rispetto alla parte destra: questo è l'ordine nella palette originale, nella maggioranza dei casi un semplice ordine numerico. Sulla destra della figura III.1 i colori sono ordinati secondo l'algoritmo di EzStego, al fine di seguire il percorso più breve attraverso il cubo RGB, percorso che corrisponde alla differenza visiva minima tra ogni valore di colore. La funzione di inserimento di EzStego lavora linea dopo linea, sulla sequenza di pixel dell'immagine a partire dall'angolo in alto a sinistra. Dopo l'embedding ogni pixel contiene un valore steganografico (cioè, un bit del messaggio). Tale valore steganografico di un pixel è rappresentato dal bit meno significativo dell'indice che quel pixel ha nella palette ordinata. La funzione di embedding confronta il valore da inserire (il bit del messaggio) con il LSB dell'indice del pixel, e sostituisce, se necessario, l'indice del pixel con il suo vicino nella tavolozza ordinata.

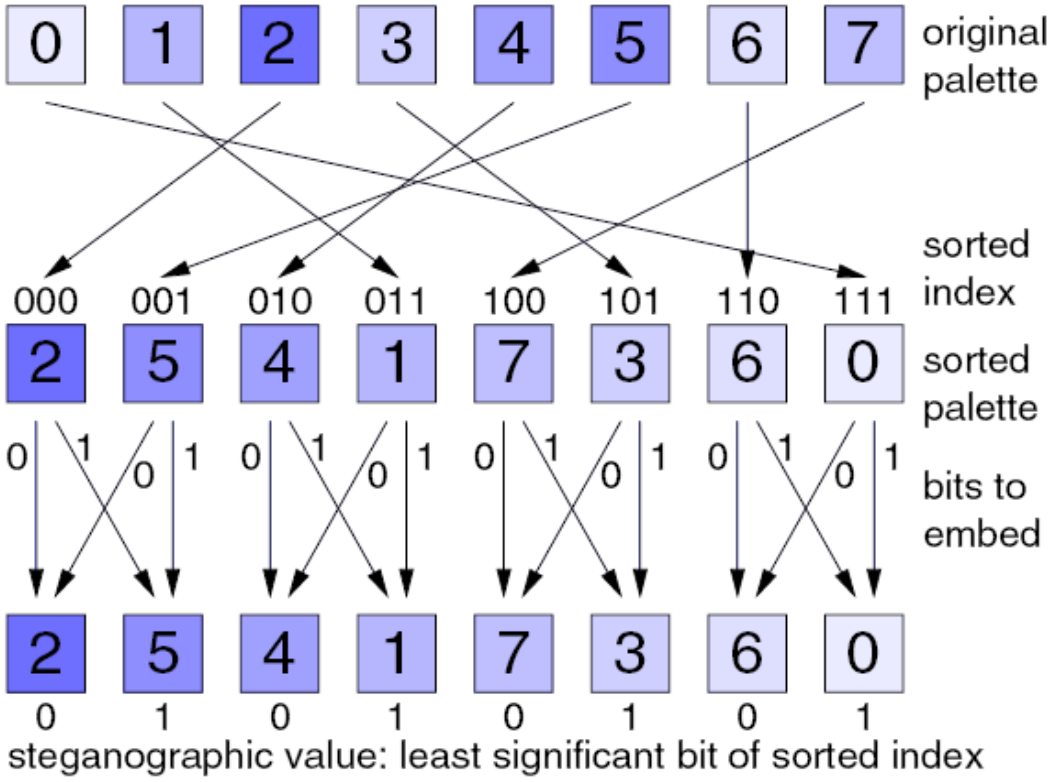


Figura III.2 – La funzione di embedding di EzStego, tratto da [19]

La figura III.2 mostra la funzione di inserimento di EzStego con una palette ridotta. Per esempio, troviamo l'indice 7 per un dato pixel nella cover image. Se vogliamo inserire un 1, basta sostituire l'indice 7 con 3, mentre se vogliamo inserire uno 0 non dobbiamo fare nulla (si osservi la parte bassa della figura III.2). Poiché il colore dell'indice 7 nella palette originale si trova all'indice 100 (=4) della tavolozza

ordinata, ed il colore di indice (non ordinato) 3 si trova all'indice (ordinato) 101 (=5), questi due colori sono molto simili, essendo adiacenti nella tavolozza ordinata e, quindi, sono visivamente quasi impossibili da distinguere. Un cambio dall'indice 7 all'indice 3 (e viceversa) è impercettibile ai nostri occhi, fino a che non si compara direttamente la stego image con l'immagine cover.
Uno dei limiti maggiori di EzStego risiede nella facile operazione (che chiunque può effettuare) di estrazione del messaggio orinale. In breve, EzStego fa steganografia ma non crittografia. Inoltre, sebbene EzStego non soffra degli attacchi di clusterizzazione esso è molto sensibile agli attacchi visuali che descriveremo nel paragrafo III.8.

III.4 Occultamento in immagini digitali

L'informazione, lo vedremo in questo paragrafo, può essere nascosta nelle immagini in molti modi diversi. Per nascondere un'informazione, l'inserimento diretto di messaggi può richiedere la codifica di ogni bit dell'informazione nell'immagine, oppure l'inserimento selettivo del messaggio in aree con rumore che attirino meno l'attenzione – quelle aree in cui ci sia una grande variazione di colori. Il messaggio può anche essere sparpagliato casualmente all'interno dell' immagine.
Gli approcci più comuni nascondere l'informazione in immagini digitali includono:

- *least significant bit (LSB) insertion* (inserimento nel LSB)
- *masking and filtering* (mascheramento e filtraggio)
- *algoritmi e trasformazioni*

ognuna di queste tecniche può essere applicata, con vari gradi di successo, a differenti file di immagini.

III.4.1 LSB insertion

L'inserimento nel LSB è un comune approccio per l'embedding di informazioni in un cover file. Sfortunatamente, questo metodo è poco robusto anche rispetto ad una leggera manipolazione dell'immagine. La tecnica base impiegata dalla maggior parte dei programmi, consiste semplicemente nel sostituire i bit meno significativi delle immagini digitalizzate con i bit che costituiscono il file segreto (i bit meno significativi sono assimilabili ai valori meno significativi di una misura, cioè quelli che tendono a essere affetti da errori). Spesso l'immagine che ne risulta non è distinguibile a occhio nudo da quella originale ed è comunque difficile dire se eventuali perdite di qualità siano dovute alla presenza di informazioni nascoste oppure all'errore causato dall'impiego di uno scanner poco preciso, ad una macchina fotografica scadente o ancora alla effettiva qualità dell'immagine originale prima di essere digitalizzata. Convertire un'immagine da un formato come GIF o BMP, che ricostruisca esattamente il messaggio originale (compressione lossless), in un JPEG,

che non lo ricostruisce esattamente (compressione lossy), e viceversa potrebbe distruggere l'informazione nascosta nei LSBs.
Per nascondere un'immagine nei LSBs di ogni byte di una immagine a 24 bit, si possono usare i 3 bytes di ogni pixel. Una immagine 1024 × 768 può potenzialmente nascondere un totale di 2359296 bits (294912 bytes) di informazione. Se si comprime il messaggio da nascondere prima di inserirlo, si può nascondere una maggiore quantità di informazione. Alla vista, la stego image risultante apparirà identica alla cover image.

Esempio III.3 *La lettera 'A' può essere nascosta in 3 pixels (assumendo che non ci sia compressione). Sia quello seguente il raster data originale per 3 pixels (9 bytes):*

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

Il valore binario di 'A' sia 1000011; volendo inserire tale valore nei 3 pixels si otterrebbe:

(00100111 11101000 11001000)
(00100110 11001000 11101000)
(11001000 00100111 11101001)

I bit sottolineati sono gli unici 3 bit realmente modificati negli 8 bytes usati. □

In media, la LSB insertion richiede che soltanto metà dei bit in un'immagine vengano modificati. In più, si possono nascondere dati sia nel LSB sia nel secondo LSB e ancora l'occhio umano non sarebbe in grado di distinguere tra stego image e cover image. La tecnica appena applicata nell'esempio III.3, rappresenta il cuore della steganografia sostitutiva, anche se di fatto ne esistono numerose variazioni.
Innanzitutto è ovvio che tutto quello che abbiamo detto vale non solo per le immagini, ma anche per altri tipi di media, per esempio suoni e animazioni digitalizzati. Inoltre – e questo è meno ovvio – lavorando con le immagini come file contenitori, non sempre si inietta l'informazione al livello dei pixel, ma si è costretti a operare su un livello di rappresentazione intermedio: è il caso, per esempio, delle immagini in formato JPEG, nel quale le immagini vengono memorizzate solo dopo essere state compresse con una tecnica che tende a preservare le loro caratteristiche visive piuttosto che l'esatta informazione contenuta nella sequenza di pixel. Se iniettassimo delle informazioni in una bitmap e poi la salvassimo in formato JPEG, le informazioni andrebbero perdute, poiché non sarebbe possibile ricostruire la bitmap originale. Per poter utilizzare anche le immagini JPEG come contenitori, è necessario lavorare su una trasformazione da valori di colore a frequenze di colore (cfr. III.4.3).

III.4.2 Masking and filtering

Le tecniche di masking e filtering, comunemente riservate per immagini a 24 bit e grayscale, nascondono informazioni marcando una immagine, in una maniera simile

al watermarking della carta. Le tecniche di watermarking possono essere applicate senza pericolo di distruzione dell'immagine dovuta alla compressione lossy poiché sono maggiormente integrate nell'immagine. Watermarks visibili non fanno parte della steganografia per definizione. La differenza è innanzitutto negli scopi. La steganografia tradizionale nasconde le informazioni; i watermarks estendono l'informazione e diventano un attributo della cover image. I watermarks digitali possono includere informazioni quali copyright, ownership o licenze, come mostrato in figura III.3.

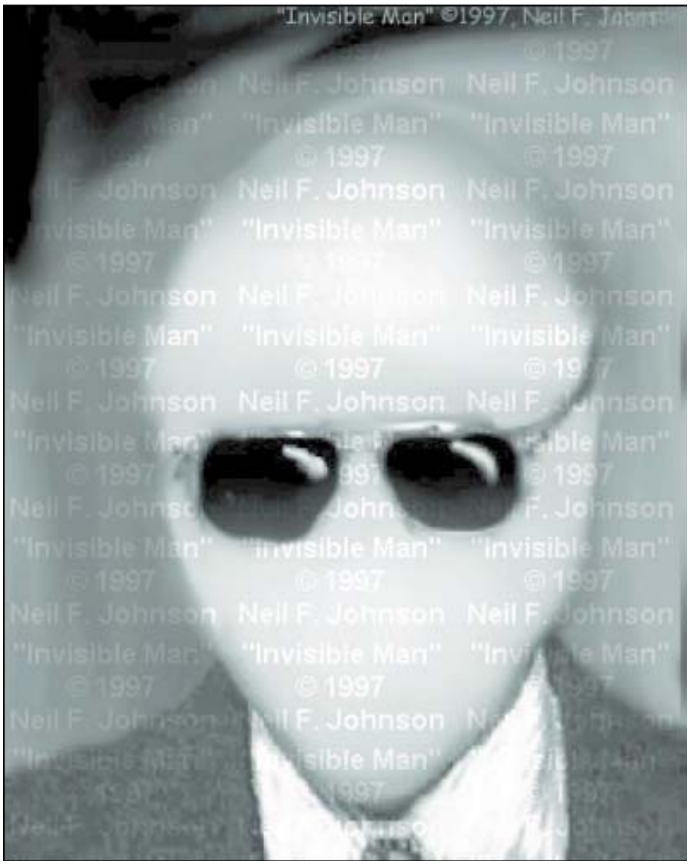


Figura III.3 – "Invisibile Man" di Neil F. Johnson

Nella steganografia, l'oggetto della comunicazione è il messaggio nascosto. Nei watermarks digitali, l'oggetto della comunicazione è la cover image.
Per creare la watermarked image in figura III.3, si aumenta la luminosità della masked area del 15%. Se si dovesse incrementare la luminosità di una percentuale inferiore, l'occhio umano non sarebbe in grado di distinguere la maschera. È ora possibile usare l'immagine con il watermark per nascondere il plaintext o informazioni codificate.
Il masking è più robusto della LSB insertion rispetto alla compressione, al cropping e ad alcune operazioni di image processing. Le tecniche di masking inseriscono informazioni in aree significative in modo che il messaggio nascosto risulti maggiormente integrato con la cover image rispetto al semplice nasconderlo in un livello "rumoroso". Ciò rende, ad esempio, il masking più adatto della LSB insertion per immagini JPEG di tipo lossy.

III.4.3 Algoritmi e trasformazioni

La manipolazione LSB insertion è un modo semplice e veloce di nascondere informazioni, ma è poco robusto rispetto a piccoli cambiamenti risultanti da image processing o compressione lossy. Questo tipo di compressione è un vantaggio delle immagini JPEG rispetto ad altri formati: immagini ad alta qualità possono essere salvate in file relativamente piccoli usando metodi di compressione JPEG; proprio per questo le immagini JPEG si sono diffuse enormemente su Internet.
Un tool steganografico che integra gli algoritmi di compressione per nascondere informazioni è Jsteg, a cui, vista la sua importanza, è dedicato il paragrafo III.6.
Le immagini JPEG impiegano la *trasformata discreta del coseno* (DCT) per ottenere la compressione. DCT non è di per sé una trasformata lossy, però lo diventa se i valori del coseno sono calcolati con precisione numerica limitata, il che introduce errori di approssimazione nel risultato finale. Le variazioni tra i valori dei dati originali e i dati calcolati dipendono dai metodi impiegati per calcolare la DCT (per i dettagli si veda il paragrafo III.5).
Anche altre proprietà dell'immagine, come la luminosità, possono essere altresì manipolate. Il patchwork e tecniche simili usano il "redundant pattern encoding" o metodi di diffusione spettrale per spargere l'informazione nascosta nelle cover images ("patchwork" è un metodo che marca delle aree dell' immagine, dette appunto patches). Questi approcci possono favorire la protezione contro l'image processing, come ad esempio il cropping e la rotazione, nascondendo le informazioni meglio del semplice masking. Inoltre, supportano la manipolazione delle immagini più prontamente dei tools basati sulla LSB insertion.
Nell'usare il redundant pattern encoding occorre effettuare un trade off fra la dimensione del messaggio e la robustezza. Ad esempio, un messaggio di piccole dimensioni può essere riportato molte volte su una stessa immagine, come mostrato in figura III.3, in modo che, se la stego image viene tagliata (cropping), c'è un'alta probabilità che il watermark resti ancora visibile. Un messaggio di grosse dimensioni, invece, può essere inserito solo una volta perché occuperebbe una porzione molto più grande dell'area dell'immagine.
Altre tecniche permettono di cifrare e disperdere (*to scatter*) i dati nascosti in un'immagine. La tecnica di scattering fa apparire il messaggio simile a del rumore. I fautori di questo approccio sostengono che anche se i bit del messaggio venissero estratti, sarebbero inutili senza l'algoritmo e la stego key per decodificarli. Ad esempio, il *White Noise Tool* è basato sulla tecnica di diffusione spettrale e dei salti in frequenza, che diffonde il messaggio attraverso l'immagine. Invece di avere *x* canali di comunicazione che sono modificati con una formula fissa ed una chiave, il White Noise Storm diffonde otto canali in un numero random generato da alcuni parametri quali dimensione della finestra di processamento e del canale dati. Ogni canale rappresenta 1 bit, cosicché ogni finestra d'immagine contiene 1 byte di informazione e molti bits inutilizzati. Questi canali ruotano, si scambiano e si sostituiscono tra loro per ottenere una diversa permutazione dei bit. Ad esempio, il bit 1 potrebbe essere sostituito dal bit 7, oppure entrambi i bit potrebbero ruotare di una posizione verso

destra. Le regole di sostituzione sono imposte dalla stego key e dai dati random della precedente finestra (in modo analogo alla codifica di DES block).
Le tecniche di scattering e di cifratura facilitano la protezione contro l'estrazione del messaggio nascosto, ma non contro la distruzione del messaggio tramite image processing. Un messaggio diffuso nei LSBs di una immagine è ancora poco robusto rispetto alla distruzione a causa della compressione lossy e dell'image processing, in quanto si tratta di un testo in chiaro inserito nei LSBs.
Il ruolo classico della steganografia nella sicurezza è quello di affiancare la crittografia, non di sostituirla. Se un messaggio nascosto è criptato, una volta scoperto deve anche essere decriptato, il che fornisce un ulteriore livello di protezione.

III.5 La discrete cosine transform

Per ogni componente di colore, JPEG usa la Discrete Cosine Transform [37], abbreviato DCT, per trasformare blocchi successivi di pixels 8 × 8 ognuno in 64 coefficienti in frequenza (che vengono chiamati coefficienti DCT).
I coefficienti $F(u, v)$ di un blocco di pixels $f(x, y)$ si ottengono tramite la seguente formula

$$F(u,v) = \frac{1}{4} C(u)C(v) \left[\sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16} \right] \quad \forall u,v \in [0,7] \quad (3.1)$$

dove $C(u)$, $C(v) = \frac{1}{\sqrt{2}}$ per $u, v = 0$; $C(u)$, $C(v) = 1$ altrimenti.

Dopo questa trasformazione, in JPEG, viene applicato l'operatore di quantizzazione

$$F^{Q}(u,v) = round\left(\frac{F(u,v)}{Q(u,v)} \right) \quad (3.2)$$

dove $Q(u, v)$ è uno dei 64 elementi di una tavola di quantizzazione, fissata a priori.
L'operazione di inversione di (3.1) avviene attraverso la formula

$$f(x,y) = \frac{1}{4} \left[\sum_{u=0}^{7} \sum_{v=0}^{7} C(u)C(v)F(u,v) \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16} \right] \forall x,y \in [0,7] \quad (3.3)$$

con $C(u)$, $C(v)$ definiti come sopra.
L'inversione (3.3), in JPEG, viene effettuata sui valori $F^{Q'}(u,v)$ dequantizzati, con

$$F^{Q'}(u,v) = F^{Q}(u,v) \times Q(u,v) \quad (3.4)$$

Poiché, a causa dell'arrotondamento, $F^{Q'}(u,v)$ è spesso diverso da $F^{Q}(u,v)$, l'inversione (3.3) non può ricostruire esattamente il risultato originario.

III.6 JSteg

Questo algoritmo, concepito da Derek Upham, è stato il primo sistema steganografico pubblicamente disponibile per immagini JPEG. L'algoritmo di embedding sostituisce *sequenzialmente* il LSB di ogni coefficiente DCT con i bit del messaggio segreto. Si tratta di un algoritmo estremamente semplice, che è riportato in figura III.4.

```
Input: message, cover image
Output: stego image
while data left to embed do
        get next DCT coefficient from cover image
        if DCT ≠ 0 and DCT ≠ 1 then
                get next LSB from message
                replace DCT LSB with message LSB
        end if
        insert DCT into stego image
end while
```

Figura III.4 – L'algoritmo JSteg

Questo algoritmo non richiede una chiave segreta, per cui chiunque conosca il sistema steganografico utilizzato è in grado di recuperare, senza alcuna difficoltà, il messaggio nascosto da JSteg (è un problema già discusso per EzStego).

III.6.1 OutGuess

OutGuess 0.1, creato da Niels Provos, modifica JSteg migliorando il passo di codifica con l'uso di un generatore di numeri pseudo-random per selezionare i coefficienti DCT da correggere. Il seme del generatore è una chiave condivisa da sender e receiver. Il LSB di ogni coefficiente prescelto viene modificato come con JSteg. Si veda la figura III.5 per i dettagli dell'algoritmo.

```
Input: message, shared secret, cover image
Output: stego image
initialize PRNG with shared secret
while data left to embed do
        get pseudo-random DCT coefficient from cover image
        if DCT ≠ 0 and DCT ≠ 1 then
            get next LSB from message
            replace DCT LSB with message LSB
        end if
        insert DCT into stego image
end while
```

Figura III.5 – L'algoritmo OutGuess 0.1

III.7 Battere la steganografia

Dopo avere esaminato alcune tecniche steganografiche di tipo sostitutivo, questo paragrafo tratta dei problemi relativi alla loro sicurezza. Prima, però, è utile ricordare che le norme che valgono generalmente per i programmi di crittografia dovrebbero essere osservate anche per l'utilizzo dei programmi steganografici. Per ciò che riguarda le specifiche caratteristiche della steganografia, si tengano presenti i seguenti principi: in primo luogo si eviti di usare come contenitori file prelevati da siti pubblici o comunque noti (per esempio, immagini incluse in pacchetti software, ecc.); in secondo luogo si eviti di usare più di una volta lo stesso file contenitore (l'ideale sarebbe quello di generarne ogni volta di nuovi, mediante scanner e convertitori da analogico a digitale, e distruggere gli originali dopo averli usati). Di questi concetti abbiamo già discusso nel paragrafo I.8.
Precisato quanto sopra, occupiamoci ora di come si può battere la steganografia.
Le tecniche sostitutive appena viste consistono nel rimpiazzare un elemento di scarsa importanza (che risulti, in sostanza, ridondante) nel file cover, con il messaggio segreto che vogliamo nascondere. Quello che viene ritenuto il principale difetto di tali tecniche è che in genere la sostituzione operata può alterare le caratteristiche statistiche del rumore presente nel media utilizzato. Un sistema steganografico è sicuro se è in grado di resistere agli attacchi steganalitici.

III.7.1 Sistema steganografico sicuro

La teoria dell'informazione ci permette di essere più specifici su cosa si intenda per sistema perfettamente sicuro. Sono stati proposti modelli effettivamente basati sulla teoria dell'informazione [36] che, come si è gia avuto modo di sottolineare (paragrafo I.5) assumono che il guardiano sia passivo.
Si assume, ancora, che l'avversario (nemico) abbia conoscenza completa del sistema di codifica, ma che non conosca la chiave segreta condivisa da sender e receiver legittimo. Il nemico deve (vuole) costruire un modello per la distribuzione di probabilità P_c di tutti i possibili cover media e per la distribuzione P_s dei possibili stego media.
L'avversario può, in questo modo, usare la *detection theory* per decidere tra l'ipotesi C (dove C sta ad indicare che lo stego object non contiene alcuna informazione occulta) e l'ipotesi opposta S (dove S indica che lo stego medium trasporta un messaggio segreto).

Definizione III.1 *Un sistema è perfettamente sicuro se non esiste alcuna regola di decisione tra le due ipotesi C ed S che sia più affidabile del semplice tirare ad indovinare.*

Precisiamo, se ce ne fosse bisogno, che sender e receiver sono in accordo sul sistema di codifica/decodifica e sulla chiave condivisa.

III.8 Attacco visuale

L'attacco visuale è uno dei più noti ed efficaci metodi per l'individuazione di contenuti steganografici all'interno di file in formati lossless. Questa tipologia di attacco, inventata da Westfeld e Pfitzmann [19], ha modificato radicalmente la teoria steganografica perché ha dimostrato che l'assunzione relativa alla casualità del bit meno significativo di un pixel (cioè che l'insieme dei LSBs possa essere paragonato a rumore) è solo un mito.
L'idea degli attacchi visuali è quella di rimuovere tutte le parti dell'immagine che coprono il messaggio segreto. L'occhio umano può, a quel punto, distinguere se c'è la presenza di un potenziale messaggio, oppure se l'immagine è innocua. La rimozione avviene attraverso un filtraggio, diverso a seconda del tool steganografico usato per l'embedding, ed ha la struttura in figura III.6:

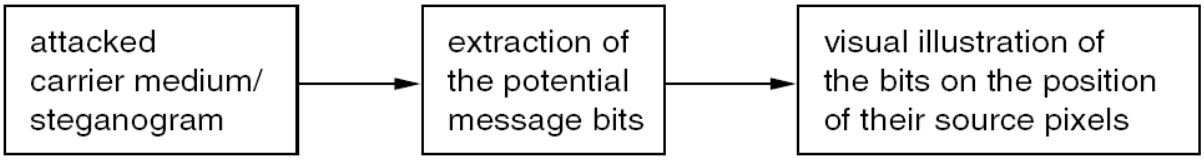


Figura III.6 – Attacco visuale, tratta da [19]

Il seguente esempio, tratto da [19], mostra come battere EzStego.

Esempio III.4 *Un filtro per l'attacco visuale presenta graficamente i valori dei pixels quando la funzione di estrazione (secondo passo in figura III.6) è applicata ad essi. EzStego usa i colori dei pixels, definiti nella palette, per determinare l'inserimento dei bit del messaggio. Il filtro che permette di attaccare EzStego sostituisce la palette originale con una bicolore (che contiene solo il bianco ed il nero). La figura III.7, a pagina seguente, illustra questa sostituzione. I colori che hanno un indice pari nella palette ordinata diventano neri, il resto diventano bianchi. I messaggi segreti che non utilizzano tutta la capacità possibile, lasciano parte del cover immodificata, poiché EzStego usa un sequential embedding. È facile riconoscere questi messaggi come si evince dalla figura III.8, che presenta nella parte alta due immagini, quella a sinistra senza alcun contenuto steganografico, quella a destra con un messaggio contenuto nella metà in alto dell'immagine. La parte bassa della figura III.8 riporta le due immagini filtrate: è chiaro quale delle due è quella con contenuto steganografico.* □

Il più grande inconveniente dell'attacco visuale risiede nell'impossibilità di automatizzare il processo di analisi, poiché esso non può essere portato avanti senza l'intervento di un operatore umano, in grado di notare anomalie nell'immagine filtrata. Un altro inconveniente consiste nella lentezza del processo di filtraggio.
Per tali motivi, è stato sviluppato l'attacco statistico, che può essere automatizzato senza problemi.

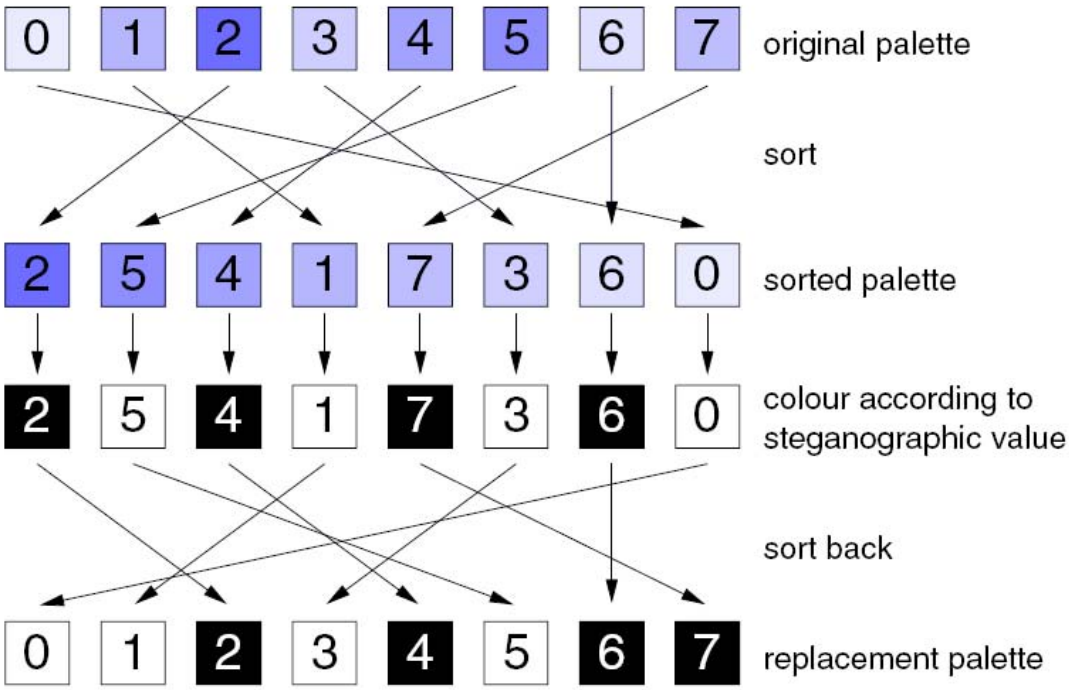


Figura III.7 – Attacco visuale ad EzStego, tratto da [19]

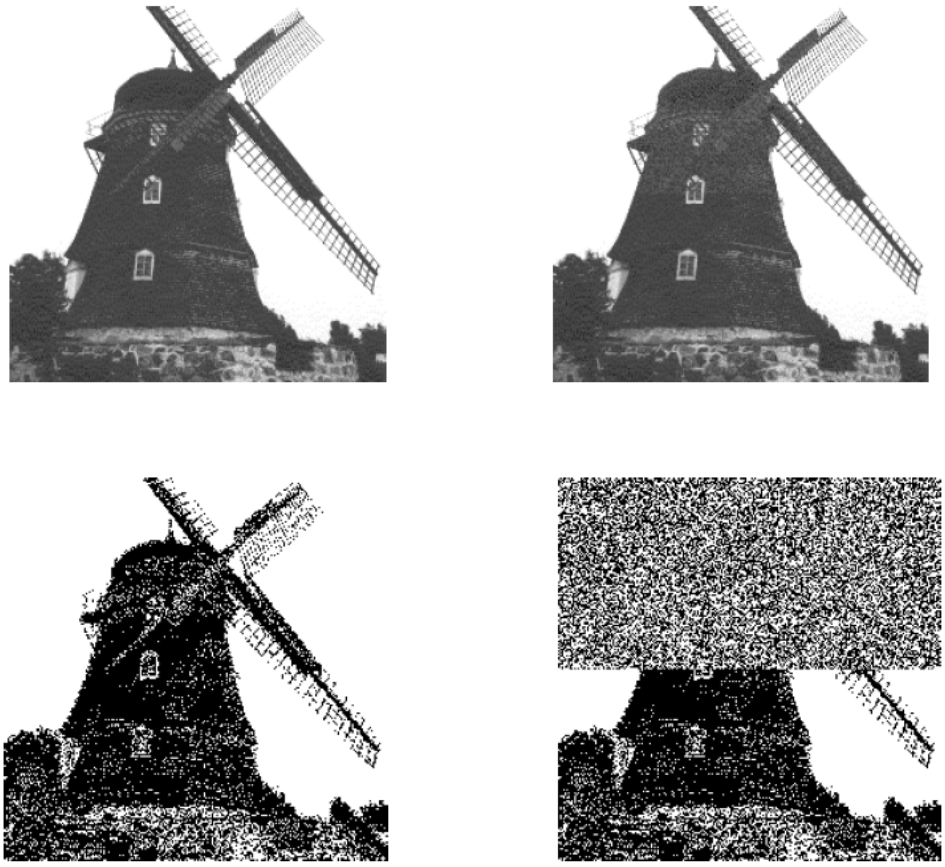


Figura III.8 – Esempio di attacco visuale, tratto da [19]

III.9 Attacco statistico

Il goal della moderna steganografia è quello di mantenere segreta la sua mera presenza, però i sistemi steganografici, a causa della loro natura invasiva, lasciano dietro di essi tracce individuabili nello stego medium. Anche se il contenuto (cioè il messaggio segreto) non è decrittato, è comunque nota la sua presenza: infatti, modificare il cover medium cambia le sue proprietà statistiche, così che un ascoltatore (avversario) può scoprire le distorsioni presenti nelle proprietà statistiche del risultante stego medium. Il processo di ricerca di tali distorsioni prende il nome di *steganalisi statistica*.
Un sistema di steganografia debole affida la propria sicurezza alla segretezza del sistema di codifica. Riprendendo l'esempio dello schiavo tatuato sul cranio (cap. I), basterebbe rasare la testa di tutti i passanti per far fallire, di fatto, quel sistema steganografico.
La stessa identica cosa vale per gli acrostici. Nell'esempio del messaggio spedito dalla spia nazista (sempre capitolo I) chiunque può estrarre il messaggio segreto, semplicemente annotando in sequenza la seconda lettera di ogni parola.
La moderna steganografia, al contrario, mira a rivelare il proprio contenuto occulto solo a chi è in possesso di una informazione segreta, che poi sarebbe la già citata chiave condivisa.
Dovrebbe essere chiaro che, ancora una volta, si tratta di rispettare il principio (sarebbe meglio dire l'insieme dei desiderata) di Kerckhoffs.
Ancora, per assicurare la sicurezza di un sistema steganografico è necessario mantenere celato il cover medium, perché se fosse esposto ad occhi indiscreti, il semplice confronto tra cover e stego object rivelerebbe immediatamente i cambiamenti apportati (si tratta della tipologia di attacco *know-cover*).

III.9.1 Steganalisi statistica su JPEG

Sebbene la steganografia sia applicabile a tutti gli oggetti che contengono informazioni ridondanti oppure rumore, in questa sezione si considerano solo immagini JPEG, perché si tratta del formato per lo scambio di immagini di gran lunga più utilizzato oggi.
In più, i sistemi steganografici per il formato JPEG appaiono più interessanti perché tali sistemi opera in uno spazio trasformato e, pertanto, non vanno soggetti ad attacchi visuali (visti nel paragrafo III.8).
Westfeld e Pfitzmann [19], ancora loro, hanno scoperto che la sostituzione sequenziale del LSB dei coefficienti DCT causa distorsioni rilevabili tramite steganalisi. L'inserimento di dati ad alta entropia (spesso dovuta alla cifratura a cui è sottoposto il messaggio prima di essere sottoposto a steganografia) cambiano l'istogramma delle frequenze di occorrenza dei colori in maniera predicibile.
Mettiamoci nel caso più semplice, in cui il passo di embedding sostituisce il bit meno significativo dei colori dell'immagine con il bit del messaggio da inserire. I colori

sono individuati dal loro indice i nella tavola dei colori; definiamo le loro rispettive frequenze di occorrenza prima e dopo l'inserimento come n_i ed n_i*.
Dato un messaggio da occultare, i cui bit siano uniformemente distribuiti, se vale la disuguaglianza $n_{2i} > n_{2i+1}$, allora i pixels che presentano il colore 2_i sono portati al valore 2_{i+1} più spesso di quanto i pixels con valore 2_{i+1} siano portati a 2_i. Come risultato, è molto probabile che valga la seguente relazione:

$$|n_{2i} - n_{2i+1}| \geq |n_{2i}^* - n_{2i+1}^*|. \qquad (3.5)$$

Semplicemente, inserire i bit di un messaggio uniformemente distribuito (cioè ad alta entropia) riduce la differenza di frequenza tra colori adiacenti.
Lo stesso ragionamento può essere esteso per i dati in JPEG. Invece di misurare le frequenze di occorrenza dei colori, bisogna osservare le differenze tra le frequenze dei coefficienti DCT. La figura III.9 è un'immagine, tratta da [4], che mostra l'istogramma delle frequenze prima e dopo che un messaggio segreto venga inserito in una immagine JPEG.

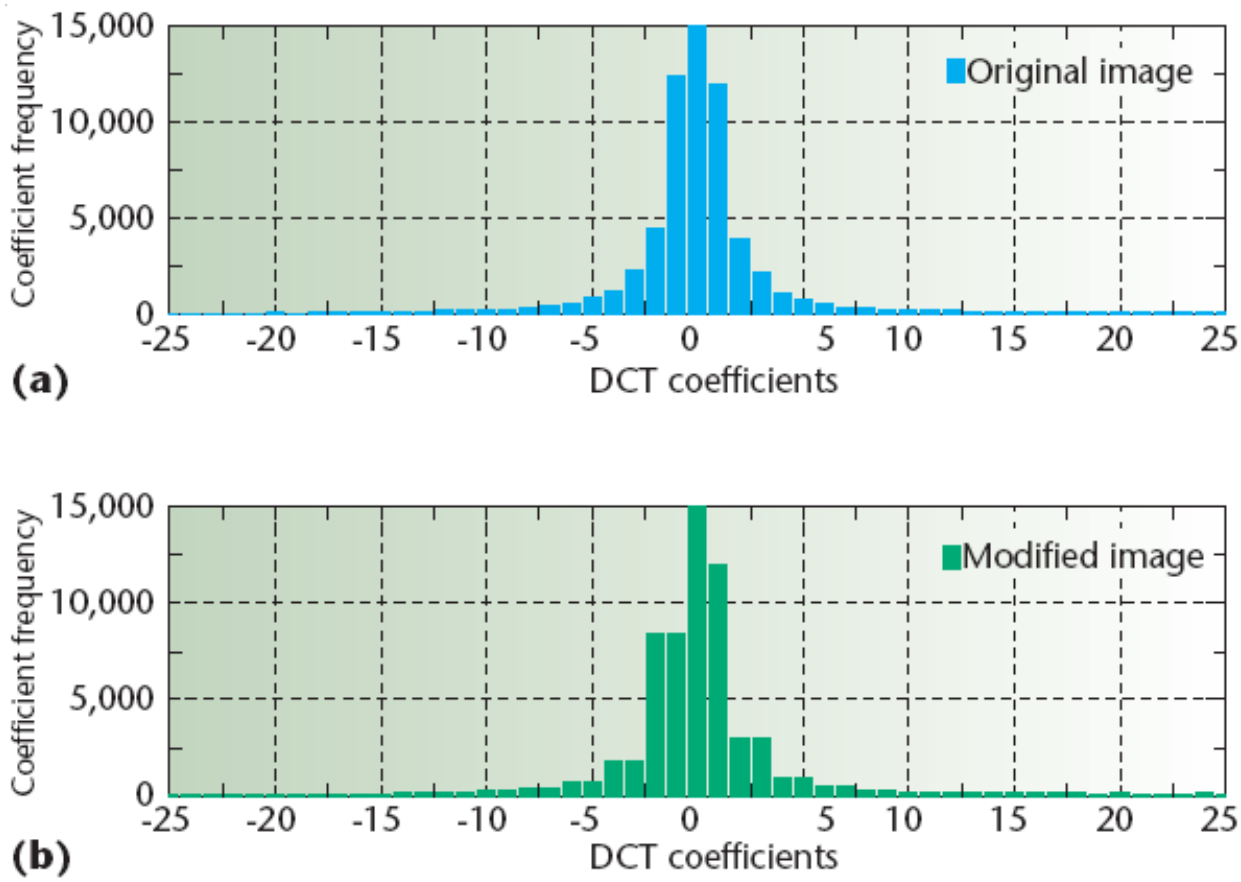


Figura III.9 – Differenze tra gli istogrammi prima e dopo l'embedding

Si nota, agevolmente, una riduzione nelle differenze di frequenza tra il coefficiente DCT -1 ed il suo coefficiente adiacente -2. La stessissima cosa si verifica tra i coefficienti 2 e 3.
Westfeld e Pfitzmann hanno usato un χ^2-test per determinare se la distribuzione di frequenza osservata y_i di una immagine corrisponda alla distribuzione attesa y_i*, che presenta una distorsione causata dall'inserimento di informazioni steganografiche.

Sebbene non sia disponibile la cover image (ci mettiamo nel caso più generale dell'attacco *stego-only*), sappiamo che la somma dei coefficienti DCT adiacenti rimane invariata tra cover e stego image. Questo ci permette di calcolare la distribuzione attesa y_i* derivante dalla stego image.
Sia n_i il totale di occorrenze del coefficiente i, calcoliamo la media aritmetica

$$y_i^* = \frac{n_{2i} + n_{2i+1}}{2} \tag{3.6}$$

per avere la distribuzione attesa e compariamola rispetto alla distribuzione osservata

$$y_i = n_{2i}. \tag{3.7}$$

Il valore χ^2 per la differenza tra le distribuzioni è dato da

$$\chi^2 = \sum_{i=1}^{\nu+1} \frac{(y_i + y_i^*)}{y_i^*} \tag{3.8}$$

dove ν sono i gradi di libertà, cioè il numero delle differenti categorie presenti nell'istogramma.
È necessario sommare valori adiacenti della distribuzione attesa e della distribuzione osservata per assicurare che ogni categoria abbia abbastanza peso. Combinare due adiacenti categorie riduce i gradi di libertà di uno.
La probabilità p che due distribuzioni siano equivalenti è data dal complemento ad uno della distribuzione cumulativa

$$p = 1 - \int_0^{\chi^2} \frac{t^{(\nu-2)/2} e^{-t/2}}{2^{\nu/2}\Gamma(\nu/2} dx \tag{3.9}$$

dove Γ è la funzione Gamma di Eulero.
La probabilità che sia presente un messaggio occultato è determinata calcolando p per un campione dell'insieme di tutti i coefficienti DCT. Il campione è costituito inizialmente dai coefficienti presenti all'inizio dell'immagine, poi per ogni misura successiva, il campione è via via incrementato. La figura III.10 mostra la probabilità che un file trattato con JSteg contenga materiale sospetto.

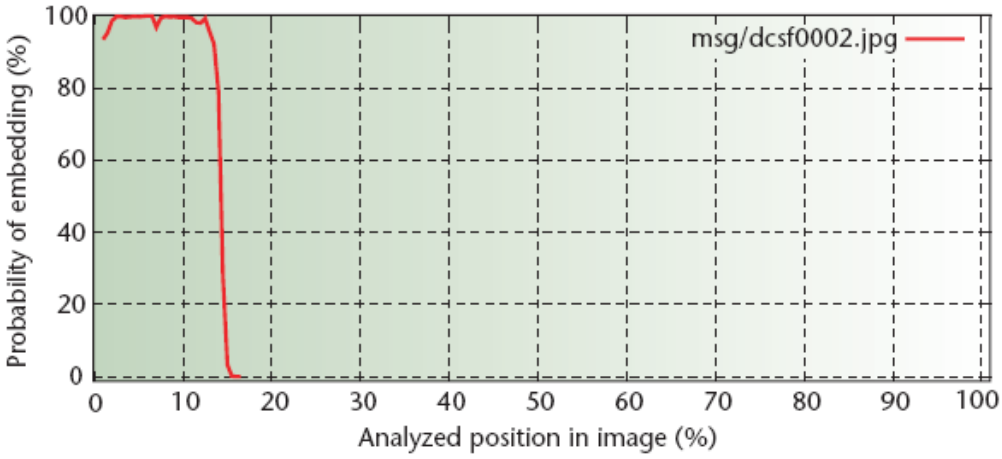


Figura III.10 – Analisi statistica di un file prodotto da JSteg

L'elevato valore per la probabilità *p* che si riscontra all'inizio dell'immagine rivela la presenza di un messaggio nascosto. Il punto in cui la probabilità decresce bruscamente indica la fine del messaggio, cioè è possibile determinare, per JSteg, oltre alla presenza del messaggio segreto, anche la sua lunghezza. Tirando le somme, l'analisi statistica, in particolare il χ^2-test, permette di attaccare con successo JSteg.
È interessante notare che il χ^2-test, che funziona ottimamente contro JSteg, non riconosce un messaggio segreto che sia casualmente distribuito su tutto il cover, quindi non riesce a battere l'algoritmo OutGuess 0.1. Comunque, è possibile estendere il test χ^2 per renderlo più sensibile alle distorsioni locali presenti in una immagine, riuscendo, così, a battere anche OutGuess 0.1. Per i particolari relativi a questa estensione si prega di far riferimento a [4].
Niels Provos ha mostrato in [40] che, applicando alcune correzioni mirate nel passo di embedding, è possibile battere la steganalisi basata sul χ^2-test. Egli ha osservato che non tutti i bit ridondanti sono usati quando si va ad inserire un messaggio. Se il test statistico per esaminare una immagine è noto, è possibile usare i bit inutilizzati per correggere le deviazioni statistiche create dal passo di inserimento. Questo tema sarà ripreso nel paragrafo V.4.

III.10 L'algoritmo F5

Questo paragrafo fornisce una breve descrizione del programma F5, sviluppato da due studiosi tedeschi, Andreas Westfeld e Andreas Pfitzmann [38].
F5 è stato sviluppato nel 2001 con lo scopo di realizzare un metodo pratico per inserire contenuti nascosti all'interno di immagini JPEG. Questo metodo avrebbe dovuto avere una alta capacità steganografica senza sacrificare la sicurezza. Guidati dal loro attacco χ^2 [19] i due ricercatori tedeschi hanno bocciato il paradigma della sostituzione del LSB nella cover image con i bit del messaggio segreto, proponendo l'incremento dei componenti dell'immagine per inserire i bit del messaggio. Invece di rimpiazzare i LSBs dei coefficienti DCT quantizzati con i valori binari del messaggio, il valore assoluto del coefficiente è decrementato di uno. Questo tipo di sostituzione non è rilevabile dal test χ^2.
L'algoritmo F5 inserisce i bit del messaggio all'interno dei coefficienti DCT quantizzati ed impiega il metodo Matrix Encoding per minimizzare il numero di cambi necessari per occultare il messaggio. F5, nella sua versione 11, accetta 6 input:

- fattore di qualità *Q* della stego image,
- file di input (TIFF, BMP, JPEG o GIF),
- il nome del file output,
- il file contenente il messaggio segreto,
- una user password da usare come seme per il generatore di numeri pseudo random (PRNG),
- il commento da inserire nell'header del file JPEG.

La fase di inserimento è riportata in figura V.4. Prima, i coefficienti DCT sono permutati attraverso il PRNG che ha la user password come seme, poi sono segmentati in gruppi di n, eliminando i coefficienti DC e quelli nulli. Il messaggio segreto viene diviso in blocchi di k bit. Per ogni blocco m del messaggio, si ottiene una parola a di un codice ad n bit concatenando il bit meno significativo del valore assoluto del coefficiente DCT in esame. Se il blocco m e il risultato della funzione di decodifica $f(a)$ sono gli stessi, allora il blocco m può essere inserito senza alcun cambiamento; in caso contrario, si usa la formula $s = m \oplus f(a)$ per determinare quale coefficiente è necessario modificare. La modifica consiste nel decrementare di uno il valore assoluto del coefficiente: se quest'ultimo diventa zero (si verifica il cosiddetto *shrinkage*), diviene impossibile stabilire in decodifica che valore steganografico si voleva inserire, perché lo zero non ha valore steganografico per F5; il coefficiente viene allora scartato, passando al prossimo coefficiente non nullo che non produca uno zero.

Input: message, shared secret, cover image
Output: stego image
initialize PRNG with shared secret
permutate DCT coefficients with PRNG
determine k from image capacity
calculate code word length $n \leftarrow 2^k - 1$
while data left to embed **do**
    get next k-bit message block
    repeat
        $G \leftarrow \{n$ non-zero AC coefficients$\}$
        $s \leftarrow k$-bit hash f of LSB in G
        $s \leftarrow s \oplus k$-bit message block
        if $s \neq 0$ **then**
            decrement absolute value of DCT coefficient G_s
            insert G_s into stego image
        end if
    until $s = 0$ or $G_s \neq 0$
    insert DCT coefficients from G into stego image
end while

Figura V.4 – L'algoritmo F5, tratto da [4]

Per maggiori dettagli su F5 si rimanda a [4, 35, 38].
È cruciale osservare che F5 non è attaccabile né con un attacco visuale, né con un attacco statistico, come dimostrato formalmente da Westfeld [35]. Questa importante proprietà di F5 sarà utilizzata per gli scopi di questa tesi, in particolare nel capitolo V.

Anticipiamo fin d'ora che sarà sviluppato e validato, nel volgere, rispettivamente, dei capitoli IV e V, un nuovo sistema steganografico con capacità di resistenza agli attacchi simile ad F5, ma che a differenza di quest'ultimo, permette anche di cifrare il messaggio con una procedura equivalente a one-time pad.

III.10.1 Attacco ad F5

Jessica Fridrich, Miroslav Goljan e Dorin Hogea hanno presentato un attacco pensato per battere F5 [4, 38]. La loro idea è consistita nello stimare l'istogramma della cover image a partire da quello ottenibile dalla stego image e comparare statisticamente questi due istogrammi, quello stimato e quello effettivo. Come risultato, essi sono riusciti ad ottenere un valore β (*modification rate*), che indica se F5 ha modificato una immagine. Il metodo di Fridrich si basa sull'analisi di come il passo di embedding di F5 vada a modificare i coefficienti AC della cover image. Sia

$$h_{uv}(d) \coloneqq |\{ F(u, v) : d = |F(u, v)|, u + v \neq 0\}| \tag{3.10}$$

il numero totale di coefficienti AC nella cover image con frequenza $F(u, v)$, il cui valore assoluto sia pari a d. $H_{uv}(d)$ è la corrispondente funzione per la stego image.
Se F5 cambia n coefficienti AC, il tasso di modifica β è n/P, dove P è il numero totale di coefficienti AC. Poiché F5 cambia i coefficienti in modo casuale, c'è da aspettarsi che i valori dell'istogramma della stego image siano

$$H_{uv}(d) < (1 - \beta)\, h_{uv}(d) + \beta\, h_{uv}(d + 1), \text{ per } d > 0; \tag{3.11}$$

$$H_{uv}(0) < h_{uv}(0) + \beta\, h_{uv}(0), \text{ per } d = 0. \tag{3.12}$$

Fridrich ed il suo gruppo hanno usato la precedente stima per calcolare il rate di modifica atteso β rispetto all'istogramma della cover image. Essi hanno trovato la massima corrispondenza, nei loro esperimenti, quando hanno usato $d = 0$ e $d = 1$, poiché sono quelli i coefficienti che cambiano maggiormente durante il passo di embedding. Tutto ciò porta all'approssimazione

$$\beta_{uv} = \frac{h_{uv}(1)\left[H_{uv}(0) - h_{uv}(0)\right] + \left[H_{uv}(1) - h_{uv}(1)\right]\left[H_{uv}(2) - h_{uv}(1)\right]}{h_{uv}^2(1) + \left[h_{uv}(2) - h_{uv}(1)\right]^2}. \tag{3.13}$$

Il valore finale di β è calcolato come la media dei β_{uv} per le frequenze

$$(u, v) \in \{(1,2), (2,1), (2,2)\}.$$

I valori dell'istogramma per la cover image sono sconosciuti e devono essere stimati a partire dalla stego image. La stima si può fare decomprimendo la stego image nel dominio spaziale. L'immagine risultante è poi tagliata eliminando quattro pixels su ogni lato per spostare l'errore ai margini del blocco. L'immagine tagliata è poi ricompressa usando la stessa tavola di quantizzazione della stego image; l'istogramma dell'immagine ricompressa è una buona stima dell'istogramma della cover image. Per tutti i dettagli si veda [38].

III.11 Sommario dei metodi e degli attacchi

Appare utile riassumere in un quadro sinottico (tabella III.1) tutte le principali tecniche statistiche analizzate in questo capitolo, collegandole agli attacchi da cui esse sono battute.

Tecnica	*Tipologia*	*Attacco*
steganografia su palette	Riduzione dei colori della palette	clusterizzazione
EzStego	LSB insertion in immagini GIF	attacco visuale
JSteg	LSB insertion in immagini JPEG	attacco statistico (χ^2-test)
OutGuess 0.1	LSB insertion in immagini JPEG + generazione random dell'ordine dei coefficienti da modificare	attacco statistico (χ^2-test localizzato)
F5	decremento dei coefficienti	attacco di Fridrich

Tabella III.1 – Quadro sinottico dei metodi steganografici più noti e degli attacchi che li battono

Capitolo IV

Steganografia e crittografia tramite stereo visione

Capitolo IV

Steganografia e crittografia tramite stereo visione

Il presente capitolo rappresenta il nucleo di questa tesi. In esso si costruiscono, passo dopo passo, le basi teoriche che portano alla definizione di un nuovo ed innovativo sistema atto a garantire sicurezza e segretezza, S^2C, che fonde insieme crittografia e steganografia tramite la stereo visione.

IV.1 Equivalenza tra mappa di disparità e file

Un file *f* (testuale, immagine, audio, video etc…) può essere visto come un insieme ordinato di byte, ogni byte può essere designato da un intero (senza segno) compreso nell'intervallo $0 \div 255$, estremi inclusi. Va da sé che *f* può essere rappresentato da un array di interi di lunghezza *L*, con *L* pari al numero totale di byte presenti in esso.
Una mappa di disparità *m*, invece, è un insieme ordinato di valori interi tra 0 e DMax, con DMax massimo valore di disparità ammesso (si veda il cap. II). Essa può essere raffigurata da un array di interi di lunghezza *M*, con *M* pari al prodotto tra ampiezza ed altezza di *m*. Se si pone DMax = 255 e, contemporaneamente, si ha $L = M$, ogni entry della mappa di disparità può rappresentare un byte, ottenendo l'equivalenza tra *m* ed *f*. Ecco trovata una corrispondenza biunivoca tra i valori della mappa di disparità ed i byte di un qualsivoglia file. La figura IV.1 illustra la relazione tra un file ed una mappa di disparità.

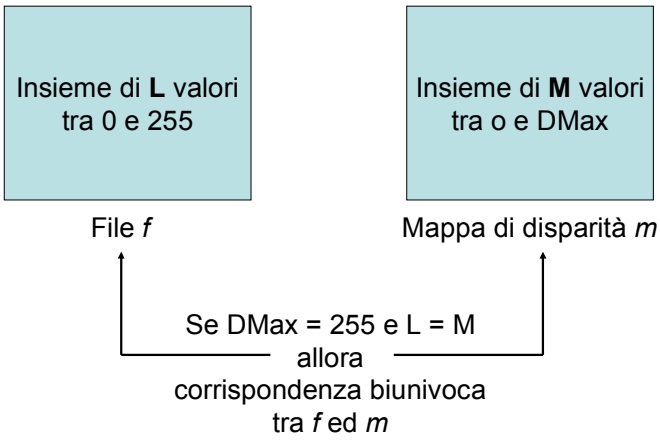


Figura IV.1 – Corrispondenza tra mappa di disparità e file

IV.2 Equivalenza tra steganografia e stereo visione

Introduciamo ora un modello generale per l'algoritmo di stereo visione visto nel capitolo II. Si tratta di un modello volutamente semplificato, al fine di rendere più chiara la trattazione seguente. La sottostante figura IV.2 rappresenta schematicamente tale modello.

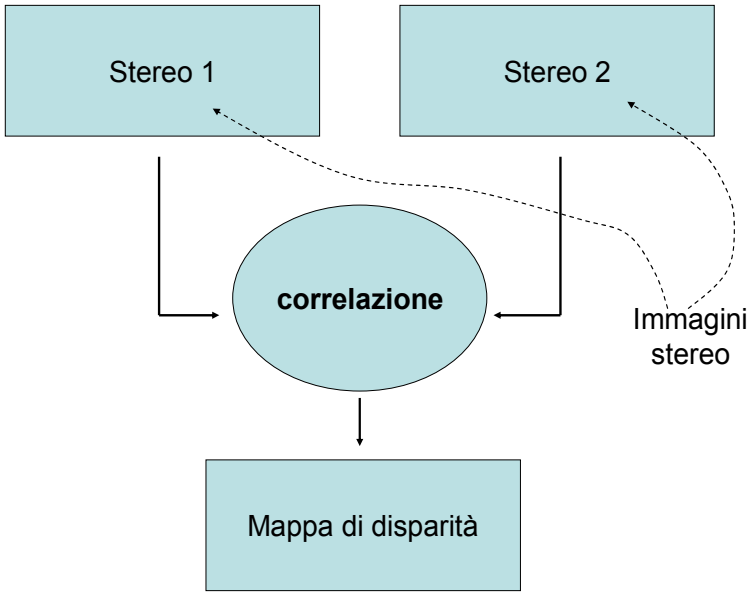


Figura IV.2 – Modello per la stereo visione

Inciso *Le due immagini stereo saranno, in seguito, denotate dai termini* stereo1 *e* stereo2*, mentre come algoritmo per la generazione della mappa di disparità si considererà la* correlazione*.*

Riprendiamo per un attimo il modello generale per la steganografia (si veda il cap. I), tralasciando momentaneamente la chiave (figura IV.3). Cosa sostituire al punto interrogativo sul lato receiver diverrà ben presto chiaro.

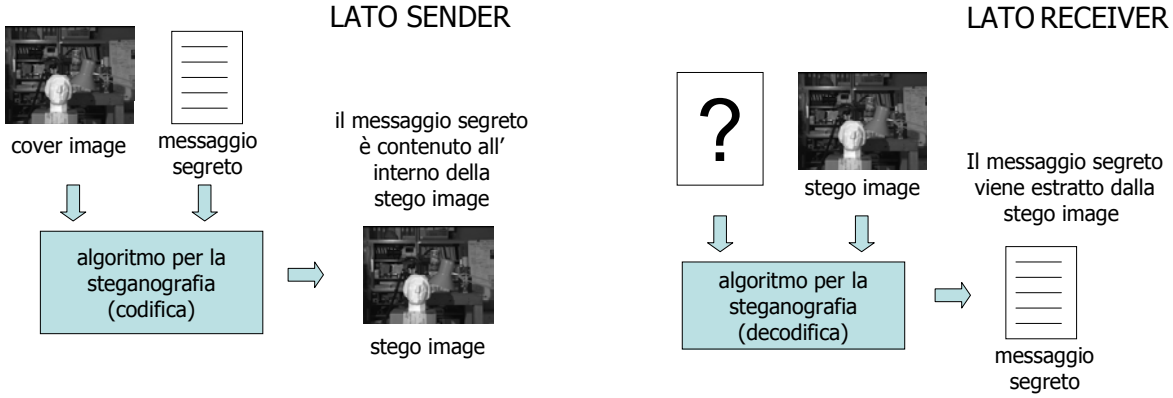


Figura IV.3 – Modello per la steganografia

Nel tentativo di mappare il modello per la steganografia sul modello per la stereo visione, ipotizziamo di far corrispondere la mappa di disparità al messaggio segreto e la cover image all'immagine s*tereo1* (figura IV.4).

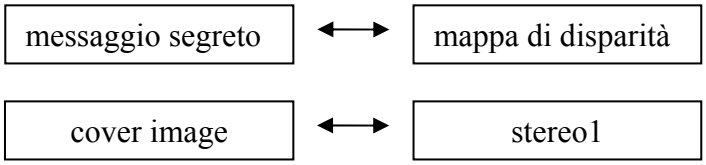


Figura IV.4 – Corrispondenze tra gli elementi dei due modelli

Per completare la corrispondenza tra gli elementi dei due modelli, rimane da assegnare la stego image. Supponendo di associare la stego image all'immagine *stereo2*, si completa il mapping tra modello per la steganografia e modello per la stereo visione (sottostante figura IV.5).

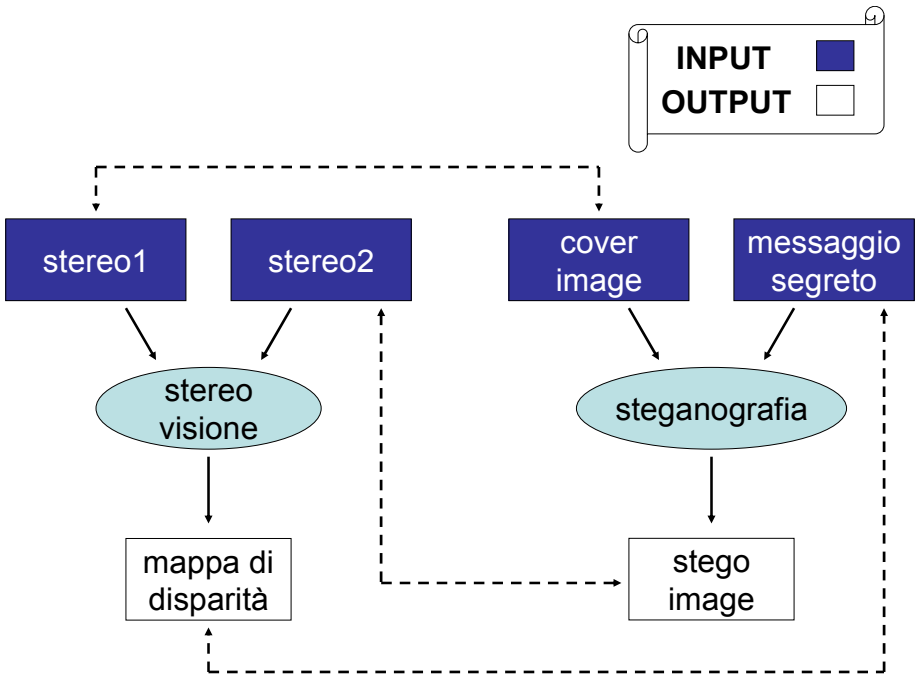


Figura IV.5 – Mapping tra i due modelli

La precedente figura evidenzia la mancanza di una esatta corrispondenza tra gli input e gli output nei due modelli. Infatti, gli input della stereo visione sono *stereo1* e *stereo2*, gli input della steganografia sono la cover-image e il messaggio segreto, ma questi 4 elementi non sono nella giusta corrispondenza: *stereo2* (che è un input) è in relazione con la stego image (che è un output) e il messaggio segreto (input) si collega alla mappa di disparità (output).
Proviamo ad ipotizzare un modello alternativo a quello della stereo visione classica, con lo scopo di preservare il corretto mapping non solo degli elementi, ma anche della loro funzione di dati in ingresso oppure in uscita (si veda la figura IV.6 nella pagina seguente).

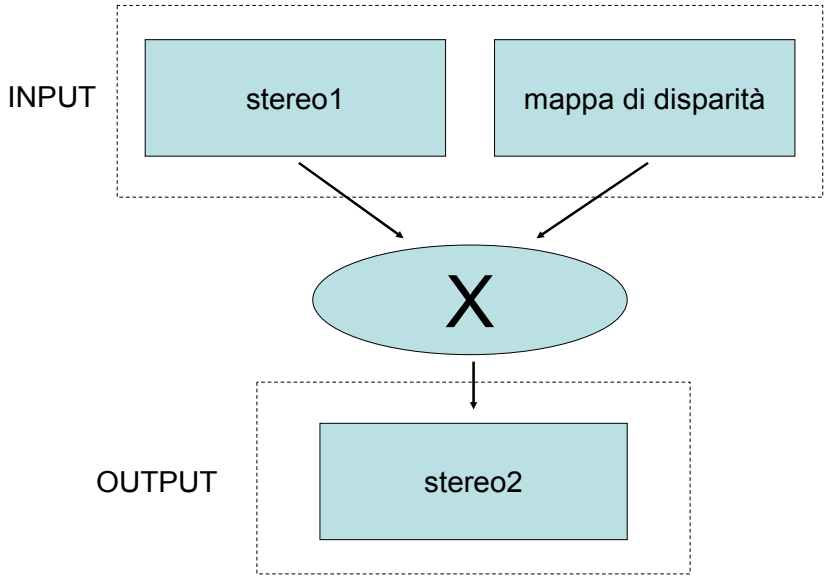


Figura IV.6 – Modello alternativo

Questo nuovo modello corrisponde per input ed output al modello per la steganografia e per componenti al modello della stereo visione (come si evince dalla figura IV.7, dove le frecce tratteggiate indicano l'esistenza di una relazione tra gli elementi dei due modelli ed il diverso colore differenzia gli input dagli output).

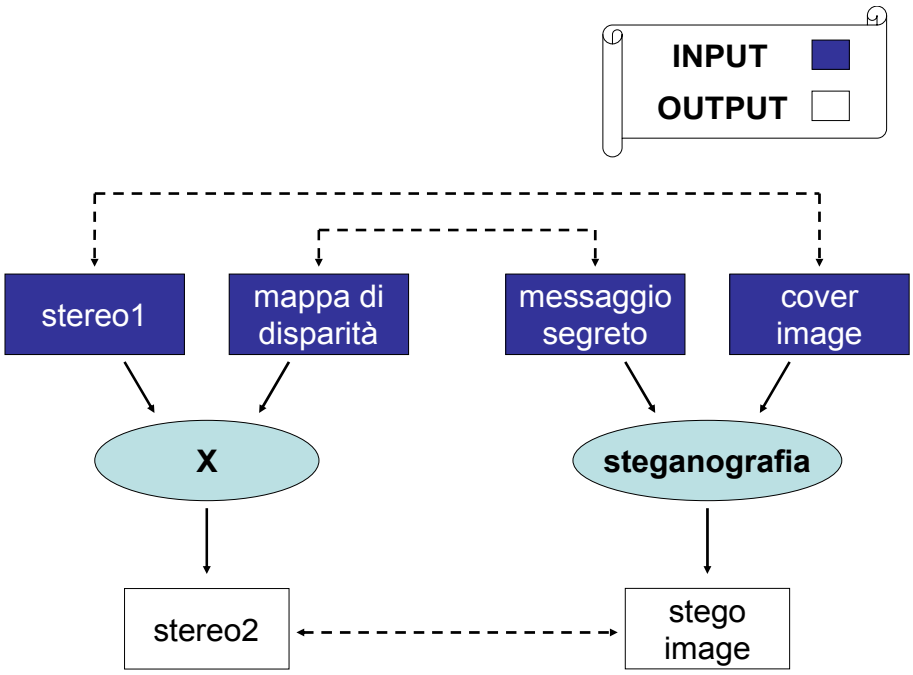


Figura IV.7 – Corrispondenza perfetta tra i due modelli

Il problema, ora, è quello di far corrispondere l'algoritmo per la stereo visione (correlazione) all'algoritmo rappresentato nelle figure IV.6 e IV.7 dalla **X**.
L'obiettivo da raggiungere diventa costruire un sistema steganografico basato su una procedura (che sarà denotata dal simbolo ~~P~~) in grado di generare l'immagine *stereo2*

(equivalente alla stego image) a partire dall'immagine *stereo1* (corrispondente alla cover image) e dalla mappa di disparità (uguale al messaggio segreto).
La procedura $\mathbb{P}$ dovrà essere perfettamente invertibile, al fine di permettere, sul lato receiver, l'estrazione del messaggio segreto (mappa di disparità) dalla stego image (*stereo2*), poiché sarà quest'ultima ad essere inviata sul canale di comunicazione.

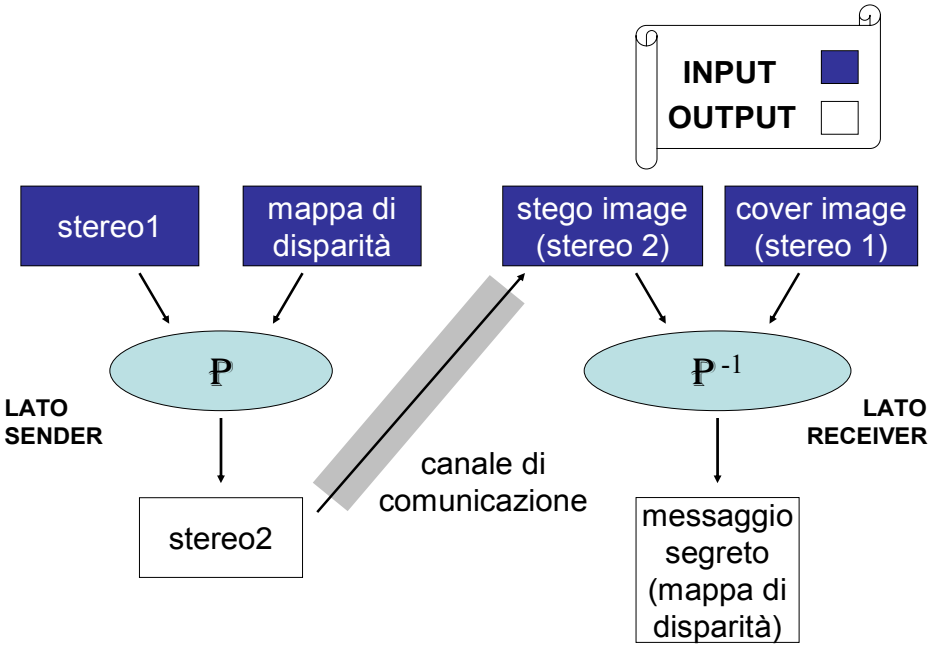


Figura IV.8 – Sistema steganografico basato sulla stereo visione

Confrontando la figura IV.2 con la parte destra (il lato receiver) della figura IV.8 si nota che $\mathbb{P}^{-1}$ corrisponde alla correlazione; resta da costruire $\mathbb{P}$, che essendo l'inverso di $\mathbb{P}^{-1}$, dovrà essere l'inverso della correlazione.

IV.3 Inversione dell'algoritmo di correlazione

Proviamo ad analizzare una possibile inversione $\mathbb{P}$ per la correlazione: si tratta di costruire l'immagine (output) *stereo2* sulla base dell'immagine (input) *stereo1*, in modo che la mappa di disparità (la quale è disponibile come input al pari di *stereo1*) risulti come soluzione della correlazione di *stereo1* e *stereo2*. In simboli *stereo2* dovrà essere costruita soddisfacendo il vincolo di eguaglianza

$$correlazione(stereo1, stereo2) = mappa\ di\ disparità \qquad (4.1)$$

Cerchiamo di spiegarci meglio, proponendo in sequenza i passi per l'inversione.

Passo 1 Scegliere una immagine. Tale immagine sarà denotata dal termine *stereo1*.

Passo 2 Trasformare il messaggio segreto che si vuole inviare in un array di interi. Una volta effettuata la trasformazione, tale array sarà individuato dal termine *messaggio*.

Passo 3 Modificare l'immagine *stereo1* in modo da ottenere l'immagine *stereo2*. Quest'ultima sarà inizialmente uguale a *stereo1*, per poi essere trasformata allo scopo di ottenere il *messaggio* come mappa di disparità tra *stereo1* e *stereo2*.

Passo 4 Verificare che il vincolo *correlazione(stereo1, stereo2) = messaggio* sia rispettato. Questo passo può essere considerato anche come fase di decifrazione, da effettuarsi all'altro estremo del canale di comunicazione (lato receiver).

Il passo 1 ed il passo 2 sono facilmente implementabili e vengono eseguiti rapidamente, così come il passo 4 (si veda a tale scopo [17]). Il passo 3 è, in linea puramente teorica, eseguibile implementando un algoritmo di correlazione inversa. Tuttavia, esistono una serie di problematiche (alcune tipiche della steganografia) che non sono facilmente affrontabili se si tenta la semplice inversione dell'algoritmo di correlazione: introduciamole.
La generazione dell'immagine *stereo2* a partire dall'immagine *stereo1*, rispettando il vincolo (4.1) imposto sulla disparità dal messaggio segreto, è un compito notevolmente complesso. Di seguito, si proverà a spiegarne il perché.
Una possibile implementazione di $\mathbb{P}$ consiste nel porre, inizialmente, *stereo1* uguale a *stereo2*, estrarre il primo valore d ($0 \leq d \leq 255$) della mappa di disparità ed assumere che la correlazione avvenga su immagini già rettificate (si veda il cap. II), con *stereo1* che va ad assumere il ruolo di immagine sinistra e *stereo2* quello di immagine destra.

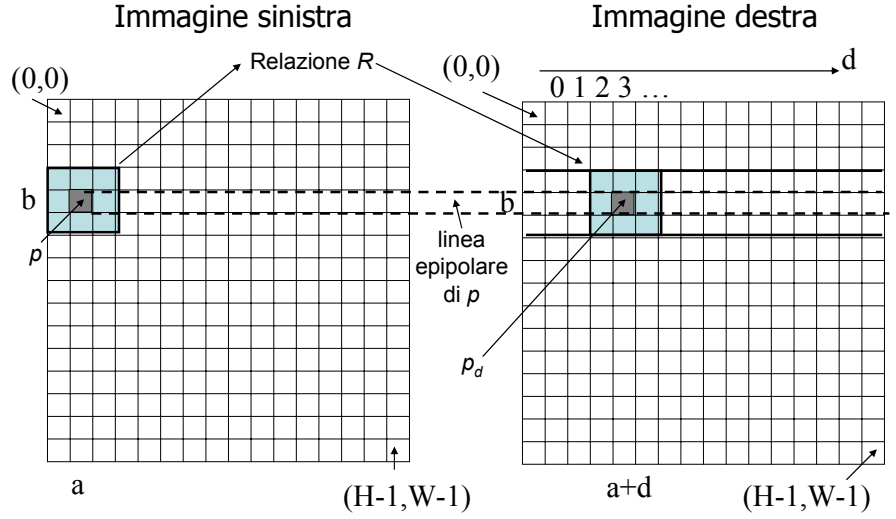


Figura IV.9 – Intorni associati tra due immagini stereo

Sia p il primo pixel di *stereo1* che deve essere processato, sia *(a, b)* la posizione di p e sia *int(p)* l'intorno di n pixels che circonda p, con p appartenente a *int(p)*. L'insieme di questi n pixels dovrà ora essere mappato su *stereo2* con disparità d. In altre parole, si dovrà creare una relazione R univoca da *int(p)* ad un intorno di *stereo2* che sia il più simile ad *int(p)* tra tutti gli intorni di *stereo2* che appartengono alla stessa linea epipolare di p, come si nota in figura IV.9 .

Affinché *R* fornisca il valore *d* come risultato, *stereo2* dovrà essere modificata in modo tale da rendere minima la disparità tra l'area intorno a *p*, che ha posizione *(a, b)* e l'area intorno al pixel p_d di *stereo2* in posizione *(a+ d, b)*; cioè si dovrà avere

$$disp(p, p_d) = min\{ disp(p, p_i) \mid p_i \text{ appartenente alla linea epipolare di } p\} \qquad (4.2)$$

Queste modifiche devono essere effettuate per tutti i pixels di *stereo1* necessari a completare il *messaggio*. Il problema di questa procedura risiede nella NON indipendenza delle modifiche che devono essere effettuate in successione su *stereo2*. Un esempio dovrebbe chiarire i concetti fin qui riportati.

Esempio IV.1 *Si ipotizzi che il messaggio da nascondere presenti i primi due byte con valori decimali, rispettivamente, $d_0 = 11$ e $d_1 = 10$. Con questi input, l'algoritmo di inversione dovrà associare la medesima area A di stereo2 ai due pixels p_0 e p_1 di stereo1. Inoltre, le modifiche necessarie per far corrispondere p_1 al pixel baricentro di A potrebbero interferire con le modifiche già effettuate al tempo in cui era p_0 l'oggetto del processamento. È necessario, ogni volta che si mette mano a stereo2 modificandola, controllare che tali modifiche non vadano ad inficiare il lavoro precedentemente effettuato (cioè che non si creino conflitti nell'area B, in figura IV.10, con le variazioni già apportate a questa zona dell'immagine destra).* □

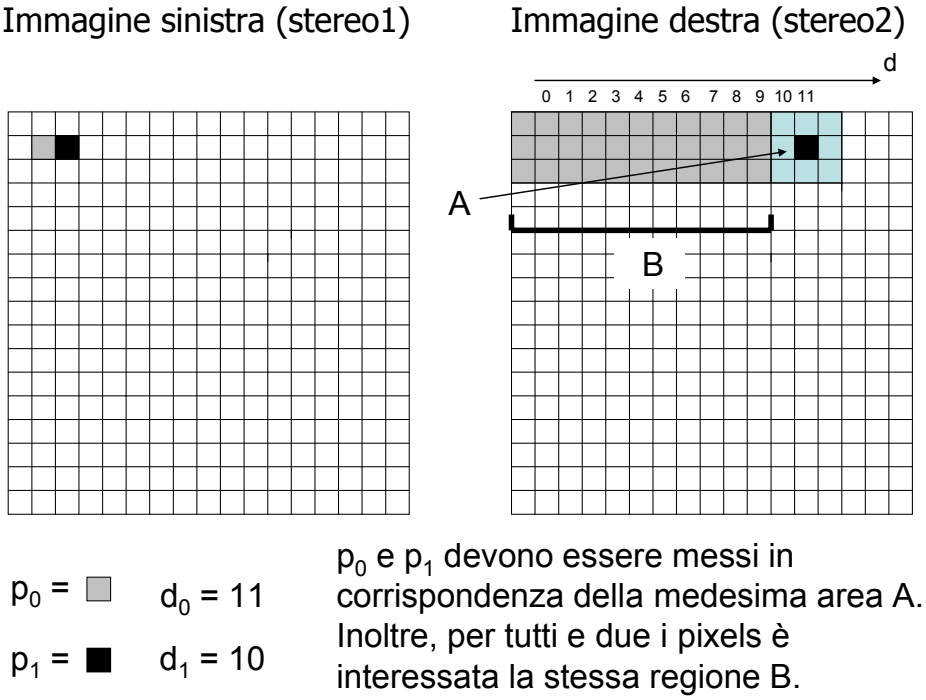


Figura IV.10 – Esempio IV.1

Lo stesso discorso, presentato nel precedente esempio IV.1, vale anche per l'elaborazione di tutti i byte del messaggio che, anche se non occupano posizioni consecutive all'interno dell'array che rappresenta tale messaggio, devono, tuttavia, essere nascosti sulla medesima linea epipolare. Infatti, la correlazione è area-based, con le aree associate ai vari pixels che possono sovrapporsi.

Inoltre, poiché l'area *A* che deve essere associata al pixel *p* che si sta processando coinvolge più linee epipolari (oltre a quella di *p*, almeno un'altra: o quella immediatamente sopra o quella immediatamente sotto la linea epipolare di *p*), ogni volta che si processa un byte estratto dall'array che rappresenta il messaggio, bisogna compiere una serie di controlli, che, seppur in numero computazionalmente trattabile, devono essere comunque effettuati per ogni nuovo byte.
Per usare un'immagine metaforica, implementare un algoritmo di correlazione (area-based) inversa sarebbe come cercare di costruire un castello di sabbia sulla battigia: ogni nuova onda (leggi ogni nuovo byte del messaggio) rischia di distruggere, almeno parzialmente, il castello (leggi lavoro già realizzato per i byte precedenti).
Se anche si riuscisse a costruire un algoritmo di correlazione inversa, sorgerebbero alcuni problemi strettamente legati alla steganografia. Per ottenere una corrispondenza biunivoca tra messaggio da nascondere e mappa di disparità, può capitare che la disparità debba assumere valori molto elevati (si è detto nel paragrafo IV.1 che DMax è uguale a 255), valori che sono molto difficili da riscontrare in *reali* (cioè, che non risultino artificiosamente ritoccate) coppie stereo. Il problema di costruire un'immagine *stereo2* che sia quanto più possibile simile ad un'immagine stereo naturale deriva direttamente dalla necessità, tipicamente steganografica, che ha *stereo2* di passare inosservata: non bisogna dimenticare che si vuole costruire un sistema steganografico (si riprenda il paragrafo IV.2). Poiché un sistema steganografico è attaccabile mediante steganalisi (par. I.9), se si va a mettere insieme un'immagine *stereo2* che sia in un qualche modo sospetta, si contravviene al principio fondante della steganografia, che è quello di mantenere segreta (di conseguenza non sospetta) la presenza di un canale di comunicazione che possa trasportare messaggi occulti.
Riassumendo, si è detto che l'inversione della correlazione area-based pone grossi ostacoli alla modifica dei valori dei pixels (elevata complessità del controllo da attuarsi su ogni singola variazione di *stereo2*) e che il valore massimo (DMax) pari a 255 provoca pesanti correzioni da effettuarsi su *stereo2* (rendendo, con buona probabilità, quest'ultima tanto diversa da *stereo1* da risultare sospetta ad un esame steganalitico). L'algoritmo invertente risulterebbe, quindi, lento e con pesanti ripercussioni su *stereo2*.
Bisogna, invece, trovare una procedura $\mathbb{P}$ semplice e veloce (si ricordino i requisiti di Kerckhoffs nel par. I.8), che generi un'immagine *stereo2* il più possibile simile a *stereo1*. In aggiunta, se ci si pone dal lato receiver, la correlazione area-based, sebbene sia piuttosto veloce [17], risulta lenta se si prova a pensare ad una sequenza di immagini da processare con un rate elevato. Oltre a $\mathbb{P}$, perciò, anche $\mathbb{P}^{-1}$ deve essere molto semplice e molto veloce.

IV.4 Una possibile procedura $\mathbb{P}$

Allo scopo di velocizzare la correlazione, $\mathbb{P}^{-1}$ potrebbe essere semplicemente un controllo pixel a pixel delle differenze tra *stereo1* e *stereo2*. Ad esempio, sia p_1 un

pixel di *stereo1* nella posizione *(a, b)* e sia p_2 un pixel di *stereo2* anch'esso in posizione *(a, b)*. Si supponga di lavorare su immagini con spazio dei colori RGB. I valori RGB di p_1 siano R_1: 127 G_1: 106 B_1: 181 e si immagini di voler nascondere la terna di bit 101. Se si va a costruire p_2 assegnando ad esso i valori R_2: 128 G_2: 106 B_2: 180, confrontando p_1 e p_2 si ottiene $R_1 - R_2 \neq 0$, $G_1 - G_2 = 0$, $B_1 - B_2 \neq 0$; un risultato che può essere interpretato, lato receiver, come 101. Dovrebbe ora essere chiaro cosa deve essere sostituito al punto interrogativo in figura IV.3: l'immagine *stereo1* (che corrisponde in termini steganografici alla cover image) deve essere presente lato receiver per poter fornire a $\mathbb{P}^{-1}$ gli input necessari al suo funzionamento. Si può, a questo punto, ultimare la figura IV.3 eliminando l'ambiguità rappresentata dal punto interrogativo e completare quel modello (figura IV.11).

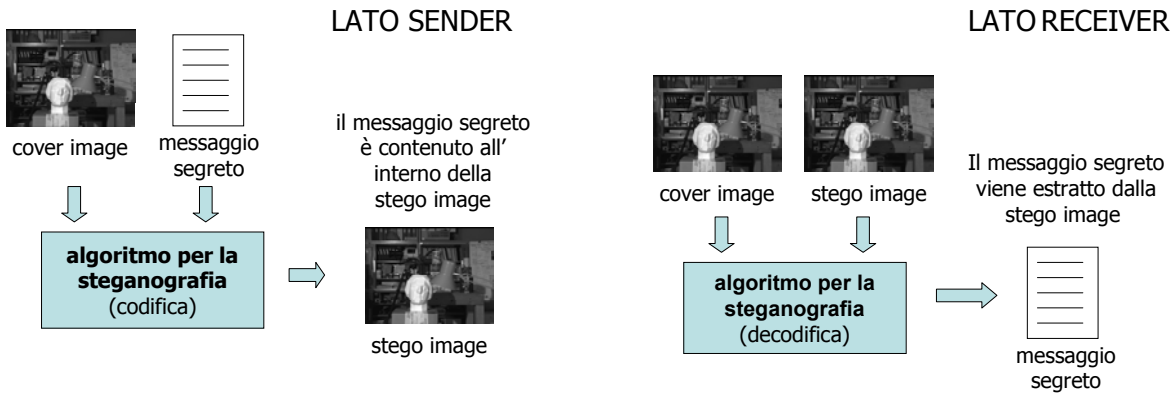


Figura IV.11 – Modello per la steganografia

Obiezione *La procedura $\mathbb{P}$, così come è stata descritta, pone un problema: il receiver deve essere in possesso dell'immagine stereo1 (cover image). Inoltre, questa procedura non è ripetibile, o meglio è ripetibile ma a rischio di destare sospetti. Difatti, l'invio ripetuto di una stessa immagine potrebbe essere facilmente registrato da una entità di controllo preposta al monitoraggio del canale di comunicazione.*

Si vedrà, in seguito, come modificare $\mathbb{P}$ per tenere conto di questa obiezione. Per ora, è interessante osservare che, in realtà, la cover image (*stereo1*) ingloba anche il ruolo di chiave del modello di steganografia riportato nella figura I.2 del capitolo I. La cover image diventa, a questo punto, **anche chiave** condivisa. Introduciamo il nuovo concetto di *key image*.

Definizione IV.1 *Si definisce* key image *l'immagine che sarà usata come chiave condivisa dell'algoritmo di cifratura/decifratura. Se il receiver non la possiede, è incapace di estrarre il messaggio segreto.*

La procedura $\mathbb{P}$ oltre ad occultare il messaggio *m* che si vuole inviare all'interno della stego image (*stereo2*), permette allo stesso tempo di criptare *m*. Questo perchè, anche se si scopre che *stereo2* contiene al proprio interno un messaggio, tale messaggio non è decifrabile a meno di possedere *stereo1* (che, si noti bene, non è inviata sul canale

di comunicazione). Se si utilizza solo una volta la cover image (che è anche key image), la cifratura è perfetta, perché la probabilità di risalire ad *m* conoscendo solo *stereo2* è la stessa di risalire ad *m* tirando ad indovinare i bit che lo compongono: si tratta della stessa idea alla base dell'algoritmo di cifratura *one-time pad* (si veda il par. I.1). Questo è possibile poiché la dimensione dello spazio delle chiavi (prodotto tra altezza e ampiezza in pixels della cover image) è maggiore della dimensione dello spazio dei messaggi (lunghezza del messaggio in bit) [Teorema I.1].
Ecco compiuta l'unione tra steganografia e crittografia tramite la stereo visione. Chiamiamo la procedura Ƥ che rappresenta tale sintesi S^2C, dalle tre iniziali di stereo visione, steganografia e crittografia espresse in notazione compatta.

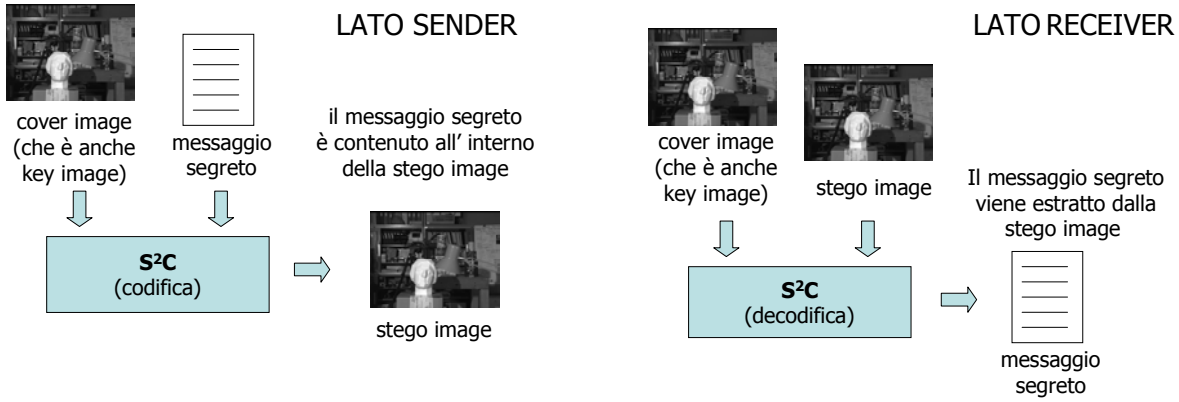


Figura IV.12 – Modello per S^2C

IV.5 L'algoritmo S^2C

Proviamo a descrivere l'algoritmo S^2C con un esempio. Sia I_1 l'immagine *stereo1* (corrispondente alla cover image) con valori RGB riportati in figura IV.13 e sia *m* il messaggio (espresso in bit) che si vuole inviare. Vogliamo costruire l'immagine *stereo2* (corrispondente alla stego image): sia I_2 tale immagine.

immagine I_1 ⟶ cover image

R: 22 G: 26 B: 51	R: 26 G: 31 B: 51	R: 32 G: 38 B: 54
R: 25 G: 29 B: 54	R: 27 G: 32 B: 52	R: 31 G: 37 B: 51
R: 30 G: 32 B: 55	R: 31 G: 33 B: 54	R: 32 G: 35 B: 50

messaggio *m*

1	1	0	0	1	1	0	1	1	0	1	0	1	1	1	0	1	1	0	0	0	0	1	1	0

Figura IV.13 – Esempio di applicazione di S^2C

Osserviamo il primo pixel in alto a sinistra di I_1, evidenziato in figura IV.13: i valori RGB risultano essere R: 22 G: 26 B: 51 mentre i primi 3 bit di *m* sono 110.
Iniziamo a costruire I_2: i colori da modificare rispetto ad I_1 sono quelli che corrispondono ai bit di *m* di valore 1. Per i bit 0 non è necessaria alcuna modifica. Naturalmente, si può anche usare una codifica inversa, modificando l'immagine in presenza di uno 0 e lasciandola inalterata se si deve occultare un 1. Nel seguito, si farà riferimento alla prima codifica, la quale appare, senza dubbio, più naturale.
La modifica dei valori RGB è random, nel senso che tali valori possono essere indifferentemente incrementati o decrementati di una o più unità. Questa libertà di modifica permette di aggirare alcune note tecniche di steganalisi che si basano sull'analisi statistica delle frequenze di occorrenza dei colori (si veda il capitolo III).
Inoltre, l'immagine I_2 è indistinguibile ad occhio nudo da I_1, poiché presenta delle differenze davvero minime rispetto a quest'ultima (si può, a ragione, definire I_2 difficilmente sospettabile). Riportiamo, per completezza, l'intera immagine I_2 ottenuta da I_1 ed *m* applicando l'algoritmo S^2C (figura IV.14).

immagine I_2 ⟶ stego image

R: 21 G: 27 B: 51	R: 26 G: 30 B: 50	R: 32 G: 37 B: 53
R: 25 G: 28 B: 54	R: 27 G: 31 B: 53	R: 31 G: 36 B: 52
R: 30 G: 32 B: 55	R: 31 G: 32 B: 55	R: 32 G: 32 B: 50

Figura IV.14 – Stego image I_2

L'inserimento dei bit avviene modificando i colori di I_1 partendo dall'angolo in alto a sinistra, per poi spostarsi in orizzontale (figura IV.15). Dopo l'inserimento dell'ultimo bit, la fine del messaggio è contraddistinta da una disparità pari a 3 unità (si veda il valore G dell'ultimo pixel di I_2 in basso a destra e lo si confronti con il corrispondente valore G di I_1).

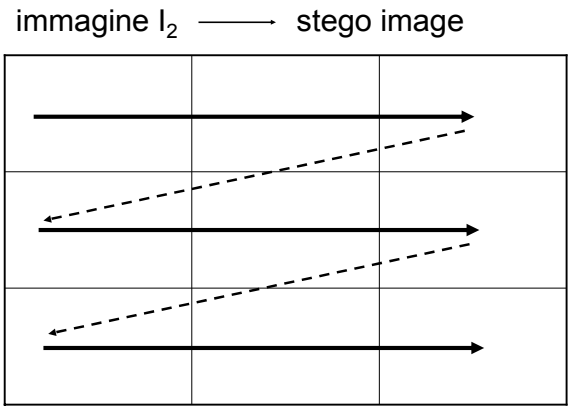


Figura IV.15 – Ordine di modifica della stego image

Il messaggio *m*, oltre ad essere stato occultato senza rendere sospetto lo stego object (obiettivo della steganografia), è stato cifrato. Infatti, se non si possiede la key image I_1, non si è in grado di risalire ad *m* (ripetiamo che si tratta di cifratura perfetta simile all'algoritmo *one-time pad* visto nel cap. I). Quindi, in un sol colpo si è ottenuta una immagine steganografica che nasconde al suo interno un messaggio criptato. Le precedenti affermazioni possono essere riassunte in due punti:

1. l'algoritmo S^2C permette di effettuare steganografia senza alterare alcuna delle principali caratteristiche statistiche di una immagine (descritte nel seguente paragrafo IV.9);
2. S^2C consente di cifrare il messaggio *m* con un algoritmo equivalente al cifrario di Vernam [22], da cui deriva one-time pad;
3. S^2C compie le azioni descritte ai precedenti punti **allo stesso tempo**.

Questi tre punti costituiscono le principali qualità di S^2C.

Obiezione *Si illustrano tali qualità senza dimostrarle.*

La risposta all'obiezione di cui sopra è la seguente: poiché la dimostrazione formale dei tre punti appena definiti è fondamentale per la validazione delle affermazioni riportate nell'interezza del presente testo, è sembrato opportuno dedicare interamente il capitolo V (a cui, caldamente, si rimanda) alla dimostrazione di essi.

IV.6 Limiti di S^2C

L'algoritmo S^2C appena presentato ha delle limitazioni. Se si usano immagini a colori (RGB), si possono nascondere un numero *N* di byte pari a

$$N = \left\lfloor \frac{W \times H \times 3}{8} \right\rfloor \tag{4.3}$$

dove *W* rappresenta l'ampiezza della cover image, *H* è l'altezza e il 3 tiene conto dei valori RGB. Se si usano immagini grayscale (in bianco e nero) il numero di byte inseribili diventa un terzo del valore *N* di sopra. Questi risultati sono in linea con le tecniche note, riportate nel capitolo III. Purtroppo, per funzionare come è stato descritto, l'algoritmo S^2C necessita di cover image e stego image in un formato *lossless*; si tratta di una pesante limitazione, poiché devono essere utilizzate immagini in formati quali bitmap, GIF oppure PNG ed, in più, se si vogliono spedire messaggi di una certa lunghezza, il prodotto $H \times W$ deve essere elevato imponendo l'utilizzo di immagini di dimensioni considerevoli. Facciamo un esempio a riguardo.

Esempio IV.2 *Sia m il messaggio, di dimensioni 110KB, che si vuole spedire. Il valore 110KB corrisponde a* $110 \times 1024 \times 8 = 901120$ *bit. Occorre un'immagine che abbia un prodotto* $W \times H$ *pari almeno a* $901120 / 3 \approx 300373$.

Se si sceglie W = 640 ed H = 480 il loro prodotto è pari alla cifra W × H = 307200: poiché ogni pixel può contenere 3 bit, si ha 307200 × 3 = 921600 < 901120, per cui una immagine 640 × 480 è adeguata ai nostri scopi. □

Il problema è che immagini in formati lossless di queste dimensioni (640 × 480) sono molto rare. Solitamente le immagini di grosse dimensioni sono compresse in formato JPEG. L'algoritmo deve, quindi, essere modificato per sfruttare immagini JPEG, come fare lo si vedrà in seguito. Ora, invece, occupiamoci di un secondo, grosso, limite di S^2C: il receiver deve possedere la key image adoperata dal sender per cifrare e nascondere il messaggio. Supponendo che sender e receiver non abbiano un archivio comune di immagini esclusive, bensì, molto più realisticamente, condividano poche di queste immagini, dovranno utilizzare sempre le stesse per inviarsi più messaggi. L'invio ripetuto delle stesse immagini può destare sospetti se il canale di comunicazione è monitorato. Dal momento che la moderna formulazione degli studi steganografici si basa sulla risoluzione del *problema dei prigionieri* (si veda il par. I.5) e Wendy monitora costantemente il canale, si deve trovare il modo di usare una sola immagine condivisa; in termini più specifici, l'algoritmo S^2C deve essere progettato per servirsi di *una sola* key image per l'invio e la ricezione di un *qualsiasi numero* di messaggi.

IV.7 S^2C a chiave unica

La dizione chiave unica indica proprio l'intenzione di utilizzare una sola key image per l'invio di più messaggi, avvalendosi di cover image diverse per ogni messaggio. Viene meno, allora, l'equivalenza tra cover image e key image che era stata prospettata alla fine del paragrafo IV.4: non si può più costruire la stego image a partire dal messaggio segreto e dalla cover image; invece è necessario introdurre un nuovo (e diverso) input per l'algoritmo di steganografia: la key image.
Per non ingenerare possibili confusioni, riportiamo le definizioni per ogni singolo termine che sarà utilizzato in seguito.

Definizione IV.2 *Si definisce* messaggio segreto *l'informazione che si vuole trasmettere sul canale di comunicazione. Deve essere nascosta la stessa esistenza di tale messaggio sul canale.*

Definizione IV.3 *Si definisce* sender *l'entità che desidera inviare il messaggio segreto (in italiano può essere tradotto dalla parola* mittente*).*

Definizione IV.4 *Si definisce* receiver *la legittima entità destinataria del messaggio segreto (in italiano può essere tradotto dalla parola* destinatario*).*

Definizione IV.5 *Si definisce* cover image *l'immagine scelta per contenere il messaggio segreto. La cover image è il contenitore vergine, cioè* prima *che sia effettuata su di esso qualsiasi modifica.*

Definizione IV.6 *Si definisce* stego image *l'immagine che contiene il messaggio segreto. La stego image è il contenitore modificato, cioè* dopo *aver effettuato su di esso le necessarie modifiche.*

Definizione IV.7 *Si definisce* key image *l'immagine che rappresenta la chiave condivisa tra il sender ed il receiver.*

Mostriamo il modello per l'algoritmo S^2C a chiave unica (figura IV.16).

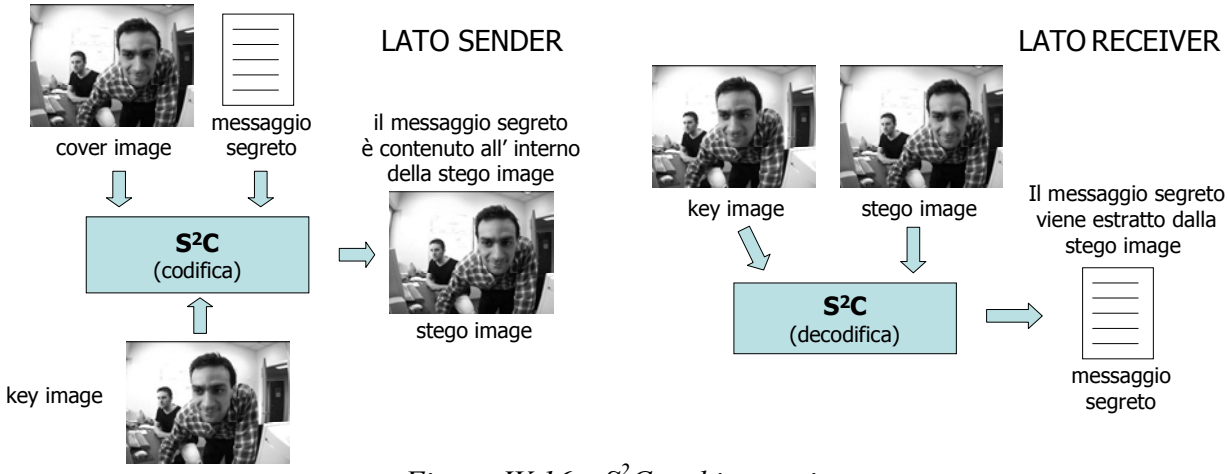


Figura IV.16 – S^2C a chiave unica

Confrontando la precedente figura IV.16 con il modello generale per la steganografia (figura I.2), si nota una perfetta corrispondenza, venendo tutti gli elementi ad essere in completa simmetria. Decade così l'ipotesi, fatta all'inizio del paragrafo IV.2, di tralasciare la chiave condivisa e non è più necessario utilizzare la cover image come input per l'algoritmo di decodifica (si veda il paragrafo IV.4).
Per permettere questi (vantaggiosi) cambiamenti, deve essere modificato l'algoritmo S^2C, sia in fase di codifica che in fase di decodifica. Si propone nel prossimo paragrafo una semplice metodica per ottenere questo scopo.
Prima di concentrarci sulle variazioni da effettuarsi su S^2C, è interessante notare che con l'introduzione della key image, rispunta idealmente l'architettura tipica dei sistemi di stereo visione. La figura IV.16 è stata proprio concepita per mostrare l'analogia tra l'architettura di S^2C e la stereo visone: cover image e key image formano una coppia stereo che è stata già mostrata nel capitolo II, paragrafo II.4.
Naturalmente, non è assolutamente necessario che cover image e stego image siano una coppia stereo, anche perché altrimenti S^2C perderebbe tutta la sua generalità risultando inutilizzabile per scopi reali.
La figura IV.16 serve ad esplicitare che senza l'intuizione permessa dalla stereo visione dell'equivalenza tra mappa di disparità e file, non sarebbe possibile costruire una trattazione teorica completa a fondamento di S^2C. Quest'ultimo può essere visto come un algoritmo di correlazione degenere, che ha un'area di riferimento pari ad un solo pixel, un solo frame di ricerca di dimensioni unitarie ed una disparità pari a zero o ad uno.

IV.8 Algoritmo a disparità comune

Sia I_c l'immagine cover e sia I_k la key image. Siano, ancora, m il messaggio segreto ed I_s la stego image. Per ipotesi, I_c ed I_k sono immagini assolutamente comuni (come ad esempio due fotografie), di uguali dimensioni $W \times H$ (con W ampiezza in pixel ed H altezza) e **diverse** tra loro per quanto riguarda i valori di colore dei loro pixel; in breve, I_c ed I_k sono due differenti immagini della stessa grandezza. Senza perdita di generalità, si assume che lo spazio dei colori di I_c ed I_k sia RGB.
Definiamo, con precisione, due coppie di pixels correlate appartenenti a I_c e a I_k.
Siano p_c un pixel di I_c, di valori R_c, G_c e B_c, che si trovi in posizione *(a, b)* e p_c' un pixel, sempre di I_c, di valori R_c', G_c' e B_c', in posizione *(a+1, b)*; dove a e b sono numeri interi, $0 \le a < W$, $a \bmod 2 = 0$ e $0 \le b < H$.
Siano p_k un pixel di I_k, di valori R_k, G_k e B_k, che si trovi in posizione *(a, b)* e p_k' un pixel, sempre di I_k, di valori R_k', G_k' e B_k' in posizione *(a+1, b)* con a e b come sopra.
Se si confrontano i valori RGB di p_c e p_c' si ottengono 3 quantità:

$$R_{dc} = R_c - R_c' \qquad G_{dc} = G_c - G_c' \qquad B_{dc} = B_c - B_c'$$

Sempre allo stesso modo, se si confrontano p_k e p_k' si ottengono

$$R_{dk} = R_k - R_k' \qquad G_{dk} = G_k - G_k' \qquad B_{dk} = B_k - B_k'$$

È molto importante notare, che, poiché esiste una certa località spaziale tra i valori RGB di pixels adiacenti in una immagine, molto probabilmente R_{dc}, R_{dk}, G_{dc}, G_{dk}, B_{dc} e B_{dk} saranno tutti valori compresi in un intervallo tra 0 e 3 o, al massimo, 4 (è comunque possibile saltare i valori che non rispettino tale intervallo). Siccome I_s deve essere costruita a partire da I_c, iniziamo col porre $I_s = I_c$; dopodichè, andremo a modificare I_s confrontando i valori di I_c ed I_k. Prima, però, definiamo con esattezza la coppia di pixels di I_s correlata alle precedenti. Siano p_s un pixel di I_s, di valori R_s, G_s e B_s, che si trovi in posizione *(a, b)* e p_s' un pixel sempre di I_s, di valori R_s', G_s' e B_s' in posizione *(a+1, b)* con a e b come sopra.
Le modifiche dovranno essere effettuate, lato sender, seguendo le regole dabbasso:

- se $X_{dc} - X_{dk} = 0$ ed il bit di m da codificare è 0 (4.4)
  allora non bisogna effettuare modifiche (*skip*)
- se $X_{dc} - X_{dk} = 0$ ed il bit di m da codificare è 1 (4.5)
  allora bisogna modificare X_s o X_s' o entrambi
  per rendere $X_{ds} - X_{dk} \neq 0$
- se $X_{dc} - X_{dk} \neq 0$ ed il bit di m da codificare è 0 (4.6)
  allora bisogna modificare X_s o X_s' o entrambi
  per rendere $X_{ds} - X_{dk} = 0$

- se $X_{dc} - X_{dk} \neq 0$ ed il bit di m da codificare è 1 allora non bisogna effettuare modifiche (skip) (4.7)

dove in tutte le regole $X = \{R, G, B\}$ e $X_{ds} = X_s - X_s'$.
Lato receiver, è possibile utilizzare una semplicissima regola per decodificare i bit di m ogni due pixels di I_s.

- se $X_{ds} - X_{dk} = 0$ allora il bit di m è 0 altrimenti il bit di m è 1 $X = \{R, G, B\}$ (4.8)

Esempio IV.3 *Siano I_c e I_k le due immagini (i cui valori RGB sono riportati in figura IV.17) che si intende utilizzare per nascondere un messaggio m, dove quest'ultimo è mostrato sotto*

m																								
	1	1	0	1	1	0	0	1	1	0	0	0	0	1	1	1	0	1	0	1	0	0	1	0

Si considerino i primi due pixels p_c e p_c' in alto a sinistra di I_c e i due pixels p_k e p_k' (coniugati di p_c e p_c') di I_k nelle medesime posizioni. I valori RGB di questi 4 pixels (evidenziati in figura IV.17) sono:

$$R_c = 89,\ G_c = 93 \text{ e } B_c = 138;\ R_c' = 88,\ G_c' = 92 \text{ e } B_c' = 137;$$
$$R_k = 36,\ G_k = 48 \text{ e } B_k = 24;\ R_k' = 36,\ G_k' = 48 \text{ e } B_k' = 24.$$

R: 89 G: 93 B: 138	R: 88 G: 92 B: 137	R: 86 G: 90 B: 135	R: 85 G: 89 B: 134
R: 89 G: 93 B: 138	R: 87 G: 91 B: 136	R: 85 G: 89 B: 134	R: 84 G: 88 B: 133
R: 91 G: 95 B: 140	R: 89 G: 93 B: 138	R: 87 G: 91 B: 136	R: 86 G: 90 B: 135
R: 96 G: 101 B: 143	R: 94 G: 99 B: 141	R: 92 G: 97 B: 139	R: 90 G: 95 B: 137

R: 36 G: 48 B: 24	R: 36 G: 48 B: 24	R: 36 G: 45 B: 26	R: 34 G: 43 B: 24
R: 38 G: 50 B: 26	R: 38 G: 50 B: 26	R: 38 G: 47 B: 26	R: 36 G: 45 B: 26
R: 41 G: 55 B: 30	R: 40 G: 54 B: 29	R: 39 G: 51 B: 29	R: 37 G: 49 B: 29
R: 43 G: 57 B: 32	R: 42 G: 56 B: 31	R: 42 G: 54 B: 32	R: 39 G: 51 B: 29

Figura IV.17 – Immagine I_c (a sinistra) e immagine I_k (a destra)

Se si confrontano i valori RGB di p_c e p_c' si ottengono 3 quantità:

$$R_{dc} = R_c - R_c' = 1 \quad G_{dc} = G_c - G_c' = 1 \quad B_{dc} = B_c - B_c' = 1$$

Sempre allo stesso modo, se si confrontano p_k e p_k' si ottengono

$$R_{dk} = R_k - R_k' = 0 \quad G_{dk} = G_k - G_k' = 0 \quad B_{dk} = B_k' - B_k' = 0$$

Poniamo inizialmente $I_s = I_c$, poi modificheremo, se necessario, i valori di colore di I_s che non rispettano le regole (4.4) ÷ (4.7). Poiché i primi 3 bit di m sono 110, dovrà essere modificato il valore di blu di uno dei due pixels di I_s, infatti:

- *primo bit di m da codificare = 1, $R_{dc} - R_{dk} \neq 0$ → SKIP – regola (4.7)*
- *secondo bit di m da codificare = 1, $G_{dc} - G_{dk} \neq 0$ → SKIP – regola (4.7)*
- *terzo bit di m da codificare = 0, $B_{dc} - B_{dk} \neq 0$ → MODIFICA I_s – regola (4.6)*

Modifichiamo, allora, il primo pixel di I_s ponendo $B_s = 137$. Si ha, in accordo con la legge (4.6),

$$B_{ds} = B_c - B_c^{'} = 137 - 137 = 0 \quad e \quad B_{ds} - B_{dk} = 0$$

Estraiamo i primi 3 bit secondo la regola (4.8)

$$R_{ds} - R_{dk} \neq 0 \rightarrow \text{il primo bit di m è 1}$$

$$G_{ds} - G_{dk} \neq 0 \rightarrow \text{il secondo bit di m è 1}$$

$$B_{ds} - B_{dk} = 0 \rightarrow \text{il terzo bit di m è 0}$$

e notiamo che la fase di decodifica è corretta. Così come sono stati inseriti i primi 3 bit di m sfruttando la correzione dei due pixel in alto a sinistra di I_s, si procede alla collocazione dei restanti bit del messaggio. Mostriamo in figura IV.18 come potrebbe apparire I_s alla fine del processamento (i valori evidenziati sono quelli modificati rispetto ad I_c). □

R: 89 G: 93 **B: 137**	R: 88 G: 92 B: 137	R: 86 G: 90 B: 135	R: 85 G: 89 **B: 133**
R: 88 G: 93 B: 138	**R: 88** G: 91 B: 136	**R: 86** **G: 90** B: 134	R: 84 G: 88 **B: 134**
R: 90 G: 95 B: 140	R: 89 G: 93 B: 138	R: 87 **G: 92** B: 136	R: 86 G: 90 B: 135
R: 95 G: 101 **B: 142**	R: 94 G: 99 B: 141	**R: 93** G: 97 **B: 140**	R: 90 G: 95 B: 137

Figura IV.18 – Immagine I_s

Le modifiche apportate nel precedete esempio non sono univocamente determinate, poiché è possibile scegliere quale dei due valori di blu può essere modificato nei primi due pixels in alto a sinistra. Infatti, alla luce dell'algoritmo steganografico a disparità comune, la precedente immagine in figura IV.18 è perfettamente equivalente alla figura IV.19 a pagina seguente:

R: 89 G: 93 B: 138	R: 88 G: 92 B: 138	R: 86 G: 90 B: 136	R: 85 G: 89 B: 134
R: 87 G: 93 B: 138	R: 87 G: 91 B: 136	R: 85 G: 89 B: 133	R: 83 G: 87 B: 133
R: 91 G: 95 B: 140	R: 90 G: 93 B: 138	R: 87 G: 91 B: 136	R: 86 G: 89 B: 135
R: 96 G: 101 B: 143	R: 95 G: 99 B: 142	R: 92 G: 97 B: 139	R: 89 G: 95 B: 136

Figura IV.19 – Immagine equivalente ad I_s

La possibilità di poter scegliere tra svariate alternative è fondamentale per rintuzzare la sempre possibile eventualità di attacchi steganalitici. Proprio variando ulteriori parti dell'immagine (oltre a quelle necessarie) possono essere inserite contromisure sia per attacchi visuali (in una certa misura) sia per attacchi statistici legati alle frequenze di occorrenza dei colori (questi attacchi possono essere annullati). Si può modificare solo un pixel della coppia interessata oppure anche entrambi; in più si può trasformare anche una coppia di valori di colore che non andrebbe necessariamente cambiata. Proponiamo una seconda immagine equivalente ad I_s che presenta aggiustamenti non strettamente indispensabili.

R: 89 G: 93 B: 138	R: 87 G: 92 B: 138	R: 86 G: 90 B: 136	R: 86 G: 90 B: 134
R: 87 G: 93 B: 137	R: 87 G: 92 B: 135	R: 85 G: 89 B: 133	R: 83 G: 87 B: 133
R: 91 G: 93 B: 138	R: 90 G: 93 B: 138	R: 87 G: 91 B: 137	R: 86 G: 89 B: 135
R: 96 G: 101 B: 143	R: 95 G: 99 B: 142	R: 92 G: 97 B: 139	R: 89 G: 96 B: 136

Figura IV.20 – Immagine equivalente ad I_s con modifiche ulteriori

Più parti di I_s possono essere modificate senza alterare la decodifica di m, più l'algoritmo di steganografia è robusto. La robustezza è un requisito molto importante per la steganografia, perché uno stego medium robusto può resistere (in una certa misura) anche agli attacchi di un guardiano attivo (si veda il par. I.5).

Obiezione *L'algoritmo a disparità comune, così come è stato presentato, è molto robusto se si vuole codificare un bit del messaggio pari ad 1 e fragile se si vuole nascondere uno 0.*

Per rispondere a questa obiezione modifichiamo le regole (4.4) ÷ (4.7) che diventano:

- ✓ se $|X_{dc} - X_{dk}| < 2$ ed il bit di *m* da codificare è 0 (4.9)
  allora non bisogna effettuare modifiche (*skip*)

- ✓ se $|X_{dc} - X_{dk}| < 2$ ed il bit di *m* da codificare è 1 (4.10)
  allora bisogna modificare X_s o X'_s o entrambi
  per rendere $|X_{dc} - X_{dk}| \geq 2$

- ✓ se $|X_{dc} - X_{dk}| \geq 2$ ed il bit di *m* da codificare è 0 (4.11)
  allora bisogna modificare X_s o X'_s o entrambi
  per rendere $|X_{dc} - X_{dk}| < 2$

- ✓ se $|X_{dc} - X_{dk}| \geq 2$ ed il bit di *m* da codificare è 1 (4.12)
  allora non bisogna effettuare modifiche (skip)

dove, come prima, in tutte le regole $X = \{R, G, B\}$ e $X_{ds} = X_s - X'_s$.

In egual modo, va modificata la legge di estrazione (4.8), che si trasforma in

- ✓ se $|X_{dc} - X_{dk}| < 2$ allora il bit di m è 0 $\quad X = \{R, G, B\}$ (4.13)
  se $|X_{dc} - X_{dk}| \geq 2$ allora il bit di m è 1

In questo modo, anche per i bit 0 da occultare si incrementa la robustezza. Si propone una immagine I_s (figura IV.21) creata applicando le nuove leggi (4.9) ÷ (4.12).

R: 89 G: 93 B: 138	R: 87 G: 91 B: 137	R: 86 G: 90 B: 135	R: 86 G: 90 B: 134
R: 88 G: 93 B: 138	R: 87 G: 91 B: 136	R: 85 G: 89 B: 134	R: 84 G: 88 B: 133
R: 91 G: 95 B: 140	R: 89 G: 92 B: 137	R: 87 G: 91 B: 137	R: 87 G: 90 B: 135
R: 96 G: 101 B: 143	R: 94 G: 98 B: 141	R: 92 G: 97 B: 139	R: 90 G: 96 B: 137

Figura IV.21 – Immagine equivalente ad I_s a robustezza superiore

IV.9 S^2C per immagini JPEG

Nel paragrafo IV.6 è stato evidenziato il più grosso limite dell'algoritmo S^2C basato su immagini lossless: per inviare messaggi di una certa lunghezza (diciamo 600KB), occorrono immagini di dimensioni superiori ad 800 × 600 pixels. Immagini lossless di tale grandezza sono estremamente rare e quindi potenzialmente sospette. Di solito, i file immagine di grosse dimensioni sono compressi con JPEG.

IV.9.1 La compressione JPEG

La compressione JPEG avviene in tre fasi, una delle quali è lossy. La prima consiste nella trasformazione in frequenza del dominio (spaziale) RGB in cui è visibile l'immagine. Questa trasformazione, perfettamente reversibile, avviene tramite la Discrete Cosine Transform (DCT, già presentata nel capitolo III). La seconda fase consiste nel dividere la matrice dei 64 coefficienti DCT per una matrice di valori prestabiliti. Questi valori sono conservati nella tabella di quantizzazione. La divisione di ogni coefficiente è arrotondata all'intero più vicino, arrotondamento che permette la compressione, ma che provoca la perdita dei dati. Infatti, il risultato della divisione è una matrice quasi completamente composta da zeri (stringhe di zeri sono facilmente codificabili con pochi bit → compressione), tuttavia quando si va ad effettuare la dequantizzazione non è possibile riottenere i valori originali, per via dell'arrotondamento (dati originali non recuperabili → codifica lossy) che non dà modo di invertire perfettamente il processo.

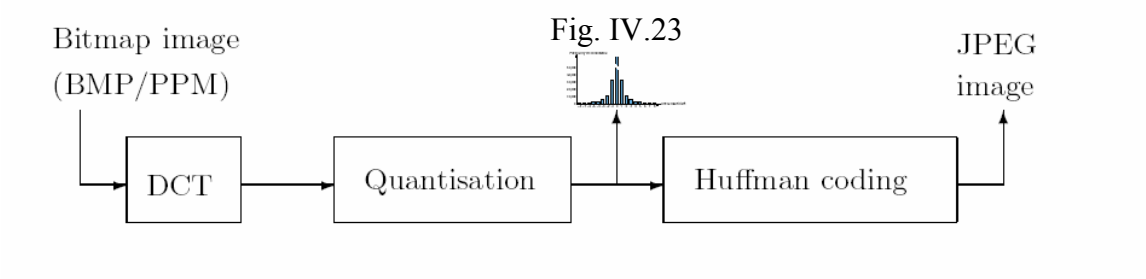


Figura IV.22 – Il flusso dei dati nella compressione JPEG, tratto da [35]

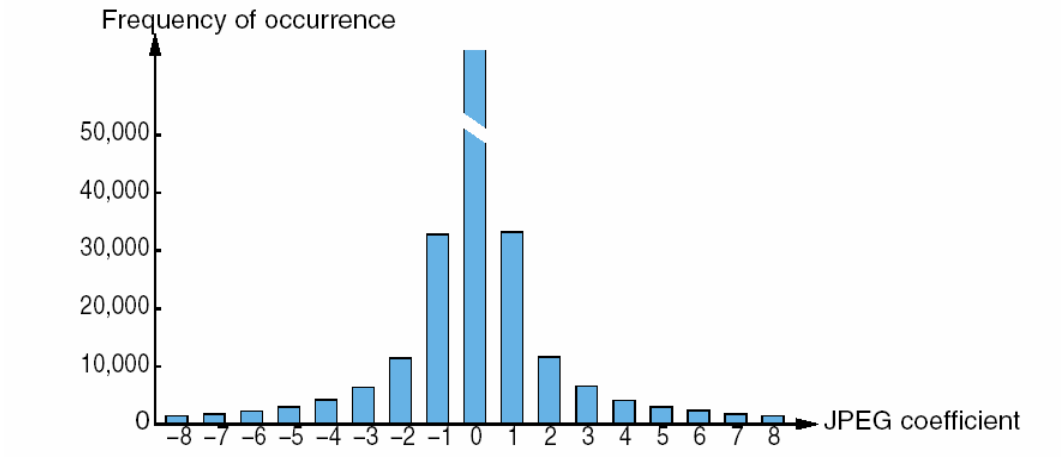


Figura IV.23 – Istogramma per i coefficienti DCT dopo la quantizzazione, [35]

La figura IV.23 mostra due importanti caratteristiche:

1. la frequenza di occorrenza dei coefficienti decresce all'aumentare del loro valore assoluto;
2. la diminuzione della frequenza di occorrenza dei coefficienti decresce all'aumentare del loro valore assoluto, cioè la differenza tra due barre dell'istogramma nel mezzo del grafico è maggiore rispetto a quella che si riscontra ai lati.

Queste caratteristiche sono proprie di ogni immagine JPEG "naturale", cioè non alterata da contenuti steganografici.

Osservazione *Un buon sistema steganografico deve produrre una stego image che preservi queste due caratteristiche.*

Dal capitolo III è noto l'attacco χ^2-test che batte senza difficoltà il sistema JSteg. Quell'attacco è possibile proprio perché l'inserimento di un messaggio segreto ad alta entropia (JSteg non cifra il messaggio, quindi quest'ultimo deve subire una pre-elaborazione crittografica che ne altera l'energia) intacca le due caratteristiche di cui sopra. A seguire, un esempio di compressione JPEG.

Esempio IV.4* *Sia la tabella in figura IV.24 un blocco 8 × 8 di valori di luminanza di una immagine. Si vuole applicare l'algoritmo di compressione JPEG a questi 64 valori.*

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

Figura IV.24 – Blocco 8 × 8 di valori di luminanza

La prima fase consiste nell'applicare la trasformata DCT al blocco precedente. La discrete cosine transform trasporta i valori di luminanza in coefficienti di frequenza, che altro non sono se non numeri reali (si veda la figura IV.25).

---

* Questo esempio è tratto da [37].

235.6	-1.0	-12.1	-5.2	2.1	-1.7	-2.7	1.3
-22.6	-17.5	-6.2	-3.2	-2.9	-0.1	0.4	-1.2
-10.9	-9.3	-1.6	1.5	0.2	-0.9	-0.6	-0.1
-7.1	-1.9	0.2	1.5	0.9	-0.1	0.0	0.3
-0.6	-0.8	1.5	1.6	-0.1	-0.7	0.6	1.3
1.8	-0.2	1.6	-0.3	-0.8	1.5	1.0	-1.0
-1.3	-0.4	-0.3	-1.5	-0.5	1.7	1.1	-0.8
-2.6	1.6	-3.8	-1.8	1.9	1.2	-0.6	-0.4

Figura IV.25 – Blocco 8 × 8 di coefficienti in frequenza (numeri reali)

Dopo la DCT, il secondo step (di quantizzazione) arrotonda adeguatamente i coefficienti di frequenza per portarli in un intervallo intero che va da -2048 a 2047. Questo secondo step è il passo lossy. La quantizzazione avviene tramite apposite tabelle che possono essere scelte arbitrariamente dal produttore dell'immagine.
Generalmente, si usano alcune tabelle (in figura IV.26 quella per la luminanza) concepite attraverso una serie di esperimenti condotti dagli sviluppatori dello standard CCIR-601, anche se non espressamente richieste dallo standard JPEG.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Figura IV.26 – Tavola di quantizzazione

Il risultato del passo di quantizzazione è riportato in figura IV.27. Si nota che dei 64 coefficienti presenti, solo 6 sono non nulli. Il coefficiente nell'angolo in alto a sinistra prende il nome di coefficiente DC, mentre le restanti 63 cifre vengono

indicate come coefficienti AC. L'usuale distribuzione dei coefficienti AC sull'intera immagine ricalca sempre la figura IV.23.
A questo punto,è possibile passare al terzo step dell'algoritmo JPEG: la codifica di Huffman dei coefficienti di frequenza quantizzati. Questo terzo passo è strettamente lossless.

15	0	-1	0	0	0	0	0
-2	-1	0	0	0	0	0	0
-1	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figura IV.27 – Coefficienti DCT dopo la quantizzazione

Invertire il processo di compressione è abbastanza semplice. Si tratta di decodificare la sequenza di Huffman per riottenere la matrice IV.27 e poi moltiplicare quei coefficienti per la tavola in figura IV.26: si ottiene un risultato pari alla figura IV.28 riportata subito sotto.

240	0	-10	0	0	0	0	0
-24	-12	0	0	0	0	0	0
-14	-13	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figura IV.28 – Coefficienti DCT dopo la dequantizzazione

Se ora si inviano i 64 coefficienti DCT dequantizzati in input alla inverse discrete cosine transform (IDCT) si ottengono i valori di luminanza ricostruiti (figura IV.29).

144	146	149	152	154	156	156	156
148	150	152	154	156	156	156	156
155	156	157	158	158	157	156	155
160	161	161	162	161	159	157	155
163	163	164	163	162	160	158	156
163	164	164	164	162	160	158	157
160	161	162	162	162	161	159	158
158	159	161	161	162	161	159	158

Figura IV.29 – Blocco 8 × 8 ricostruito, valori di luminanza

I valori ricostruiti (figura IV.29) sono molto simili ai valori originali (figura IV.24), sebbene non perfettamente uguali. □

IV.9.2 Steganografia su JPEG

Le tecniche note di steganografia agiscono modificando i coefficienti AC dopo la quantizzazione, poiché è possibile ricostruirli senza perdite nella fase di decompressione.

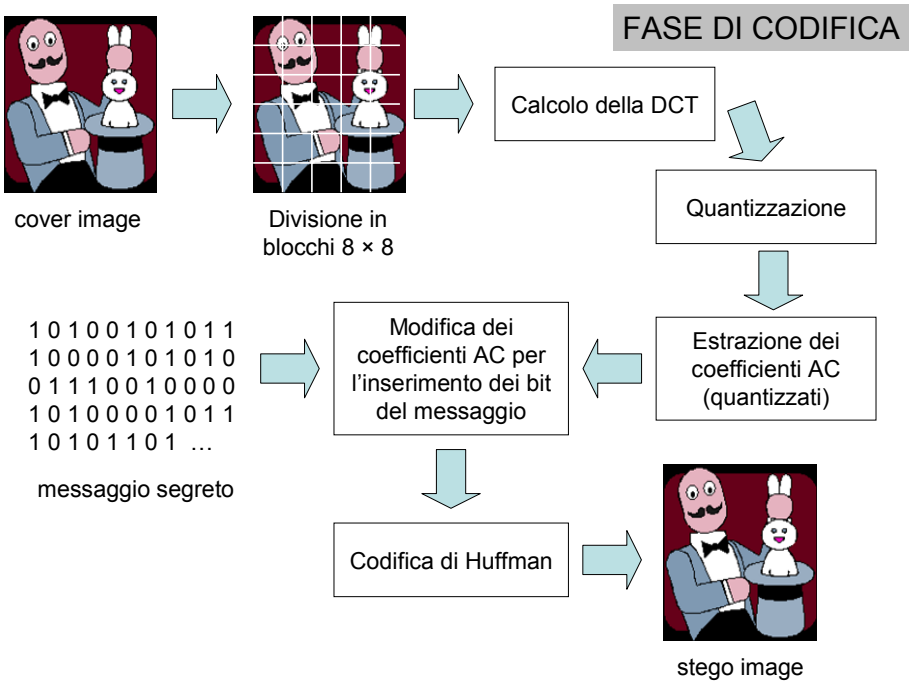


Figura IV.30 – Steganografia su immagini JPEG, fase di codifica

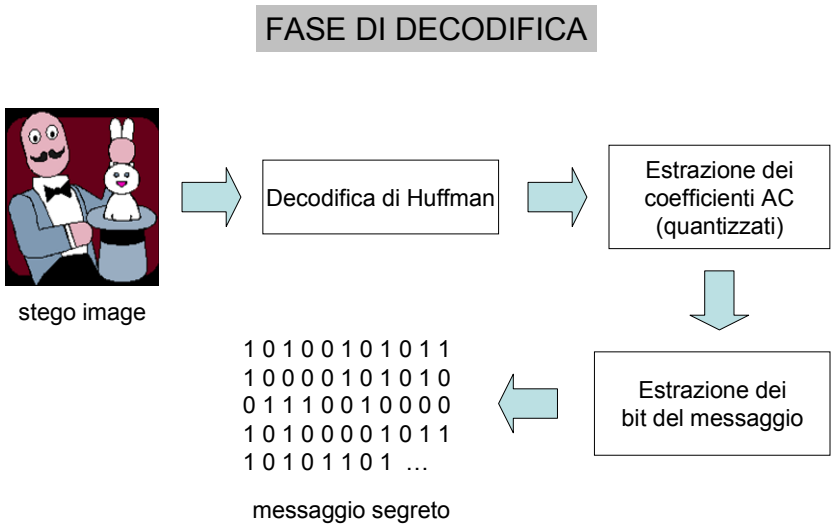


Figura IV.31 – Steganografia su immagini JPEG, fase di decodifica

I due schemi precedenti mostrano le consuete operazioni che effettuano i sistemi steganografici attualmente conosciuti (tra cui JSteg, OutGuess ed F5). Proponiamo, affinché risultino evidenti le differenze, la coppia di schemi che deve implementare il sistema S^2C.

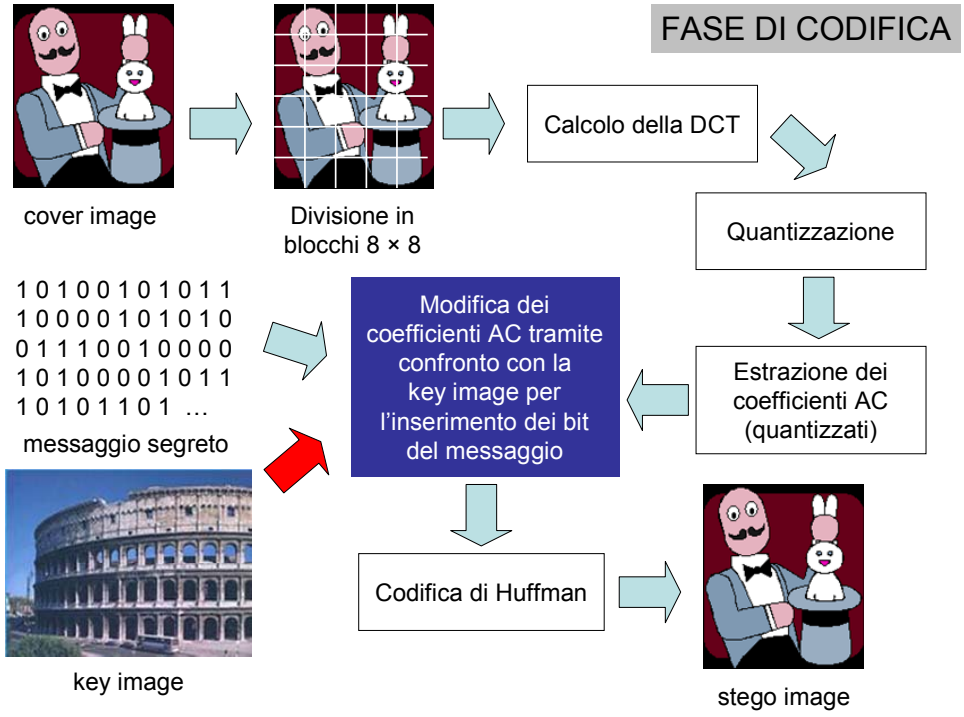


Figura IV.32 – S^2C per immagini JPEG, fase di codifica

Il confronto tra la figura IV.30 e la figura IV.32 fa emergere le due caratteristiche peculiari della codifica tramite S^2C: la presenza di una immagine chiave che viene utilizzata per criptare il messaggio segreto ed il meccanismo di modifica dei

coefficienti AC (fase steganografica) che ancora una volta si avvale della key image e non si limita a sostituire il bit meno significativo di ogni coefficiente con i bit del messaggio.

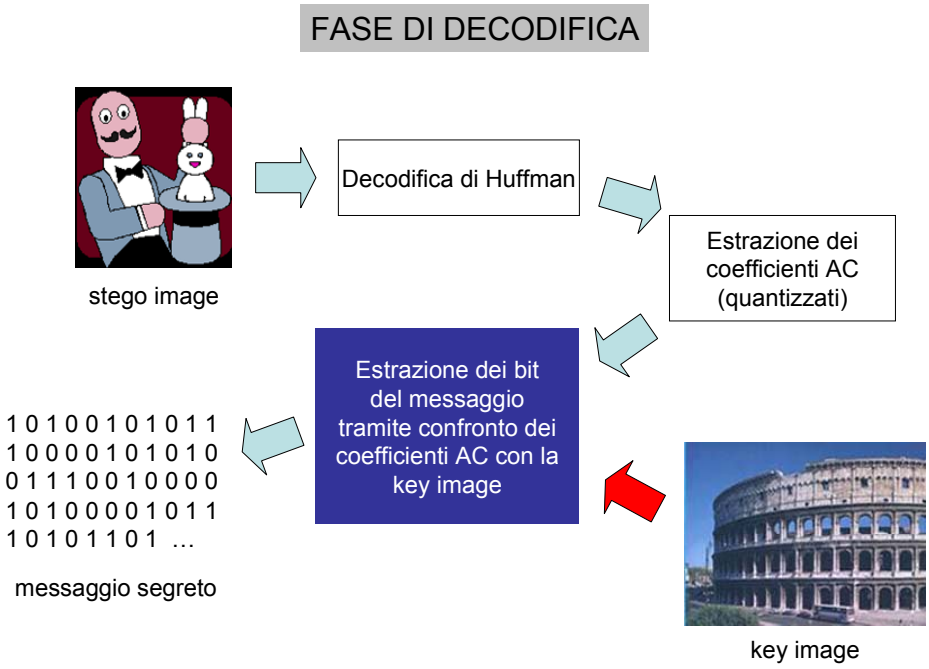


Figura IV.33 – S²C per immagini JPEG, fase di decodifica

La fase di decodifica dell'algoritmo S²C (figura IV.33) differisce dalla decodifica classica (figura IV.31) per la presenza della key image che oltre a permettere l'estrazione dei bit, **allo stesso tempo**, ne permette la decifrazione. Vediamo come avviene in dettaglio questo strano "gioco di prestigio".

IV.9.3 S²C su JPEG

Affinché S²C possa essere applicato ad immagini JPEG è necessario seguire tutti i passi dell'algoritmo riportato in basso (fase di codifica):

Passo 1C Estrazione dei coefficienti AC NON NULLI della cover image e loro inserimento in un array di interi (segnati) che si indicherà, in seguito, con coverAC[ ].

Passo 2C Estrazione dei coefficienti AC NON NULLI della key image e loro inserimento in un array di interi (segnati) che si indicherà, in seguito, con keyAC[ ].

Passo 3C Trasformazione del messaggio segreto in un array di bit che si indicherà, in seguito, con message[ ].

Ciclo Fino a che non si termina di trattare tutti i bit del messaggio, ripeti 4C e 5C.

Passo 4C Estrazioni del bit in posizione *i* del messaggio segreto (cioè message[*i*]), del coefficiente coverAC[*i*] e del coefficiente keyAC[*i*].

Passo 5C Modifica del coefficiente coverAC[*i*] secondo la seguente routine

if message[*i*] = 1 //si deve codificare un 1
then
    if coverAC[*i*] e keyAC[*i*] sono entrambi pari od entrambi dispari
    then incrementa o decrementa coverAC[*i*] *
    end if
    else non fare nulla
    end else
end if
else //si tratta del caso message[*i*] = 0, si deve codificare uno 0
    if coverAC[*i*] e keyAC[*i*] sono uno pari ed uno dispari
    then incrementa o decrementa coverAC[*i*] *
    end if
    else non fare nulla
    end else
end else

** se coverAC[i] è uguale a 1 può solo essere incrementato, invece se coverAC[i] è uguale a -1 può solo essere decrementato. Si deve evitare di produrre coefficienti nulli che non sarebbe possibile estrarre in fase di decodifica.*

Passo 6C Costruzione della stego image sfruttando l'array coverAC[ ], che è stato via via MODIFICATO tramite incremento e decremento delle sue entries. Indicheremo con stegoAC[ ] l'array coverAC[ ] modificato.

Occupiamoci ora della fase di decodifica; si tratta di un algoritmo estremamente semplice e veloce.

Passo 1D Estrazione dei coefficienti AC NON NULLI della stego image e loro inserimento in un array di interi (segnati) stegoAC[ ] che sarà uguale a quello definito nel passo 6 della fase di codifica.

Passo 2D Estrazione dei coefficienti AC NON NULLI della key image e loro inserimento in keyAC[ ] che sarà uguale all'array definito nel passo 2 della fase di codifica.

Passo 3D Creazione di un array di bit che si indicherà, in seguito, con message[ ]. La dimensione massima di message[ ] sarà pari a

$$min\ \{length(\ stegoAC[\ ]\ ),\ length(\ keyAC[\ ]\ )\ \}$$

Ciclo Fino a che non si termina di decodificare tutti i bit del messaggio ripeti i passi 4D e 5D. Il ciclo viene ripetuto un numero di volte pari alla lunghezza dell'array che contiene i bit del messaggio.

Passo 4D Estrazioni del bit del coefficiente stegoAC[*i*] e del coefficiente keyAC[*i*].

Passo 5D Decodifica del bit *i* del messaggio message[*i*] secondo la seguente routine

if stegoAC[*i*] e keyAC[*i*] sono o entrambi pari o entrambi dispari
then message[*i*] = 0;
end if
else message[*i*] = 1;
end else

Alla fine del ciclo diventa disponibile in output il messaggio originale.
Mostriamo l'uso degli algoritmi di codifica e decodifica tramite un esempio molto semplificato.

Esempio IV.5 *Sia coverAC[ ] il seguente array di interi segnati [-2, -1, -1, 1, -1, -2] e sia keyAC[ ] uguale a [4, 1, 1, -1, 3, 2]. Si vuole codificare il messaggio, espresso in bit, [0, 1, 1, 1, 0, 0]. Applichiamo l'algoritmo di codifica:*

coverAC[ ]	*keyAC[ ]*	*messaggio[ ]*	*coverAC[ ]'*
-2	*4*	*0*	*-2*
-1	*1*	*1*	*-2*
-1	*1*	*1*	*-2*
1	*-1*	*1*	*2*
-1	*3*	*0*	*-1*
-2	*2*	*0*	*-2*

Otteniamo un array coverAC[ ]' modificato rispetto all'originale coverAC[ ] che chiamiamo stegoAC[ ]. Applichiamo l'algoritmo di decodifica:

stegoAC[ ]	*keyAC[ ]*	*messaggio[ ]*
-2	*4*	*0*
-2	*1*	*1*
-2	*1*	*1*
2	*-1*	*1*
-1	*3*	*0*
-2	*2*	*0*

□

Esempi concreti di applicazione di S^2C per immagini JPEG saranno mostrati nel capitolo V.

Nota importante *Da questo momento in poi, quando si parlerà di S^2C, si farà sempre implicito riferimento all'algoritmo S^2C su JPEG per due motivi:*

1. *l'uso esclusivo di immagini JPEG non riduce minimamente la portata generale delle considerazioni che seguono, poiché è possibile dimostrare l'equivalenza teorica tra S^2C su immagini lossless ed immagini lossy. Basta effettuare una trasformazione tra dominio dei valori RGB e dominio dei coefficienti di*

frequenza, possibile poiché i due domini di definizione sono mappabili tra loro;

2. *la scelta del formato immagine ha un grande impatto sul progetto di un sistema steganografico sicuro. Formati semplici e non compressi, come BMP, se da un lato forniscono molto spazio per effettuare steganografia, dall'altro la loro ovvia ridondanza li rende automaticamente sospettabili. Infatti, alcuni studiosi [38, 40] non considerano nemmeno questi formati per fini steganografici poiché sostengono che lo scambio di immagini non compresse è "equivalente" all'uso della crittografia, cioè si tratta di file facilmente riconoscibili come lo sono i messaggi cifrati (esempio I.2).*

Capitolo V

Risultati originali. Confronto con lo stato dell'arte

Capitolo V

Risultati originali. Confronto con lo stato dell'arte

Questo capitolo contiene le dimostrazioni delle tre, fondamentali, affermazioni fatte alla fine del paragrafo IV.5, che si possono riassumere così: S^2C compie una cifratura forte del messaggio segreto (par. V.3.1), S^2C fa steganografia senza alterare le caratteristiche statistiche del cover medium (par. V.3.2), S^2C realizza le precedenti operazioni allo stesso tempo (nota conclusiva in V.6).

V.1 Crittografia + steganografia*

Si è precedentemente accennato all'eventualità che i dettagli di funzionamento di un sistema steganografico possano cadere nelle mani del nemico (si veda il paragrafo I.5, dove si è supposto che Wendy sia al corrente dell'algoritmo che Alice e Bob potrebbero, eventualmente, utilizzare per nascondere il loro messaggio). In ambito crittografico si danno le definizioni di vari livelli di robustezza di un sistema, a seconda della capacità che esso ha di resistere ai vari tipi di attacco: in particolare, i sistemi più robusti sono quelli che soddisfano i requisiti posti dal *principio di Kerckhoffs*. Quest'ultimo, riformulato (si veda il paragrafo I.7) in ambito steganografico, suona più o meno così:

Principio di Kerckhoffs in ambito steganografico *la sicurezza del sistema deve basarsi sull'ipotesi che il nemico abbia piena conoscenza dei dettagli di progetto e di implementazione del sistema stesso; la sola informazione di cui il nemico non può disporre è una sequenza (corta) di numeri casuali - la chiave segreta - senza la quale, osservando un canale di comunicazione, egli non deve avere neanche la più piccola possibilità di verificare che sia in corso una comunicazione nascosta.*

Se si vuol aderire a questo principio, è evidente che le tecniche (F5 escluso) esposte nel capitolo III non sono ancora soddisfacenti per caratterizzare un sistema steganografico completo. Infatti, se i dettagli di implementazione dell'algoritmo sono resi di dominio pubblico, chiunque è in grado di accedere ad eventuali informazioni nascoste, semplicemente applicando il procedimento inverso (ad esempio, è molto semplice "riaggregare" i bit meno significativi di una stego image trattata con tecniche sostitutive oppure estrarre il messaggio segreto da contenitori "palettizzati").

---

* la quasi totalità di questo paragrafo è tratto da [21]

In altre parole, tali tecniche non permettono di cifrare il messaggio segreto, ma solo di occultarne la presenza; inoltre, la sicurezza steganografica di questi sistemi è legata alla segretezza del processo di inserimento dei bit del messaggio, mentre la sicurezza crittografica è nulla, poiché il messaggio, una volta scoperta la sua presenza, può essere ricostruito, senza difficoltà, **identico** a come appariva prima dell'occultamento nel cover medium.

Inciso *La possibilità di conoscere con precisione i particolari progettuali di un sistema steganografico (o crittografico) è essenziale se si vuole incrementarne la sicurezza. Il processo iterativo che comprende le due macrofasi di progettazione e pubblicazione prima, di analisi ed attacco poi, permette di evidenziare i difetti del sistema, che potrà essere riprogettato in modo da risultare più sicuro di quanto lo fosse in precedenza. Questo processo iterativo (in figura V.1), è realizzabile solo se gli apparati steganografici (e crittografici) sono liberamente disponibili e consultabili. Come regola generale,* si deve sempre diffidare da sistemi che basino la loro sicurezza su processi di codifica e decodifica che hanno la necessità di restare segreti.

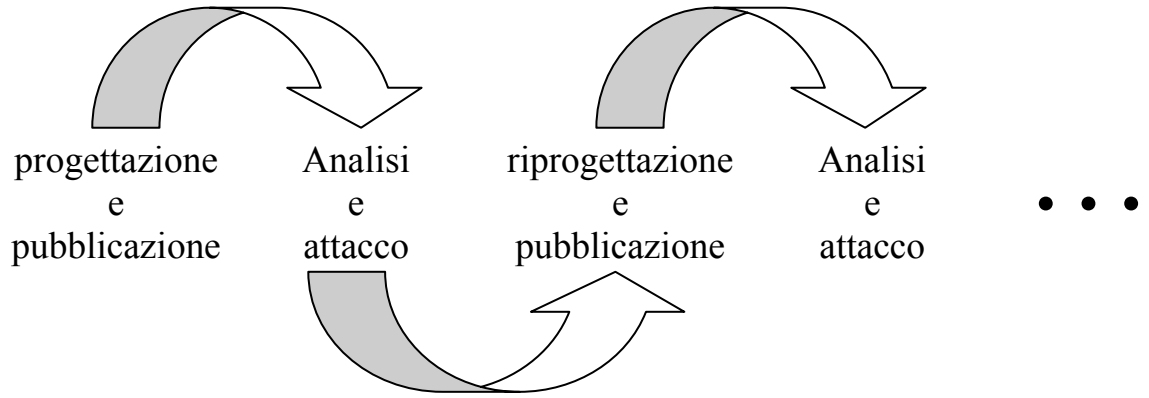


Figura V.1 – Processo di sviluppo di un sistema steganografico o crittografico

Per affrontare i problemi summenzionati (legati a sistemi steganografici che non effettuano cifratura del messaggio segreto) è necessario introdurre una fase di pre-elaborazione (figura V.2) del file segreto, che lo renda non riconoscibile - da parte del nemico - come portatore di informazioni significative. La soluzione più ovvia è quella di impiegare un sistema di crittografia convenzionale (come PGP, si veda l'esempio I.2), il quale garantisce che il nemico non abbia accesso al messaggio vero e proprio, cioè il messaggio denominato *plaintext* in figura V.2.

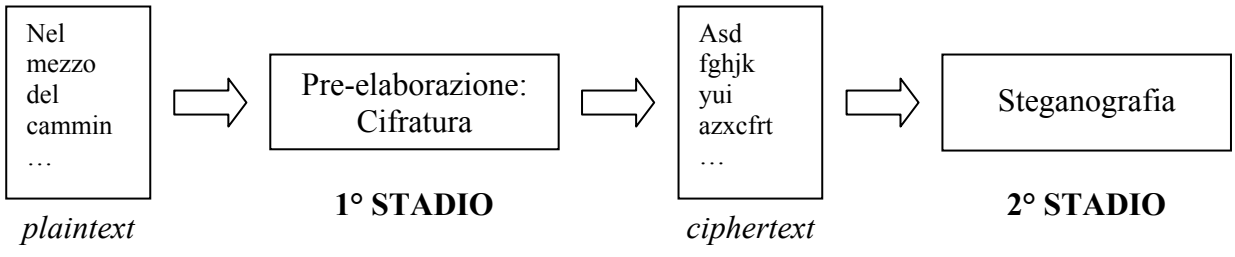


Figura V.2 – Pre-elaborazione del messaggio segreto

Questa soluzione, purtroppo, non è completa perché, in un *meccanismo a due stadi* di tal genere, il secondo processo è reversibile: chiunque può estrarre il file costituito dalle informazioni che fluiscono dal primo al secondo stadio (chiamato *ciphertext* in figura V.2). Poiché si presume che un crittoanalista esperto possa facilmente riconoscere un file prodotto da un programma di crittografia convenzionale (si faccia riferimento all'esempio I.2), lo schema in figura V.2 è da considerarsi insufficiente.
Questo punto (la riconoscibilità) è di importanza fondamentale, perché rende definitivamente non valido il sistema steganografico, indipendentemente dal fatto che il contenuto dell'informazione segreta resti inaccessibile.
Di fatto, mentre il progettista di un algoritmo di crittografia assume che il nemico impiegherà tutte le risorse possibili per decrittare il messaggio, il progettista di un sistema steganografico deve supporre, invece, che il nemico tenterà di rilevare la sola esistenza del messaggio. In altri termini, la crittografia fallisce il suo scopo quando il nemico legge il contenuto del messaggio; la steganografia, invece, fallisce quando il nemico, semplicemente, si rende conto che esiste un messaggio segreto dentro il file contenitore, pur non potendolo leggere.
È opportuno, quindi, che il messaggio crittografato, prima di essere immerso nel contenitore, venga "camuffato" in modo da diventare difficilmente distinguibile da semplice rumore. A questo scopo, sono stati escogitati diversi metodi. Il più semplice è quello di eliminare dal file cifrato da PGP tutte le informazioni che lo identificano come tale: il PGP, infatti, genera un file che rispetta un particolare formato, contenente, oltre al blocco di dati cifrati vero e proprio, informazioni piuttosto ridondanti che facilitano la gestione del file da parte dello stesso PGP (o di shell in grado di trattare con questo formato). Esiste un piccolo programma, Stealth, capace di togliere – e di reinserire nella fase di ricostruzione – tutte le informazioni diverse dal blocco di dati cifrati. Il file che esce da Stealth appare come una sequenza di bit del tutto casuale, che è molto difficile distinguere da rumore ad alta entropia. Naturalmente, chiunque può provare ad applicare il procedimento inverso (prima Stealth per ricostruire l'intestazione, quindi il PGP), ma solo disponendo della chiave giusta si potrà alla fine accedere al messaggio in chiaro. In caso contrario non si potrà neppure capire se il fallimento sia dovuto al fatto di non disporre della chiave giusta oppure, verosimilmente, al fatto che l'immagine non contiene alcun messaggio nascosto.
Un metodo alternativo all'uso congiunto di PGP e Stealth è dato dall'uso di programmi espressamente progettati per trasformare un file in rumore apparente (per esempio, *White Noise Storm*, a cui si è accennato nel capitolo III).
Riassumendo, un sistema steganografico completo deve prevedere due fasi fondamentali:

1. trasformazione del messaggio in chiaro in rumore apparente. Questa fase comprende l'uso di un sistema di crittografia e, quindi, di un qualche tipo di chiave;

2. iniezione nel (o generazione del) messaggio contenitore.

V.2 Caratteristiche di S^2C

Il precedente paragrafo ha mostrato come, solitamente, si tenda a progettare sistemi dedicati alla steganografia. Questi sistemi sono incompleti, poiché non forniscono nessuna protezione per il messaggio segreto.
Anche il meccanismo a doppio stadio risulta imperfetto perché semplicemente si limita a spostare in avanti, nel processo di codifica, il problema di fondo: è sempre possibile per un avversario ottenere il messaggio che è stato inserito nel cover medium. Se si riesce ad ottenere questo messaggio ed anche se quest'ultimo è cifrato, si riesce a violare il sistema steganografico, perché un messaggio (cifrato o non) rispetta alcuni standard di formato che possono essere riconosciuti.
La soluzione prospettata alla fine del paragrafo V.1 prevede l'inserimento di un ulteriore stadio di processamento, cioè l'uso di un programma che ripulisca il messaggio cifrato da quelle caratteristiche che lo renderebbero facilmente identificabile.
Evidentemente, tutti questi accorgimenti che devono essere osservati, complicano un pochino il sistema steganografico. Infatti, per occultare un messaggio dovremmo attraversare i seguenti stadi (figura V.3):

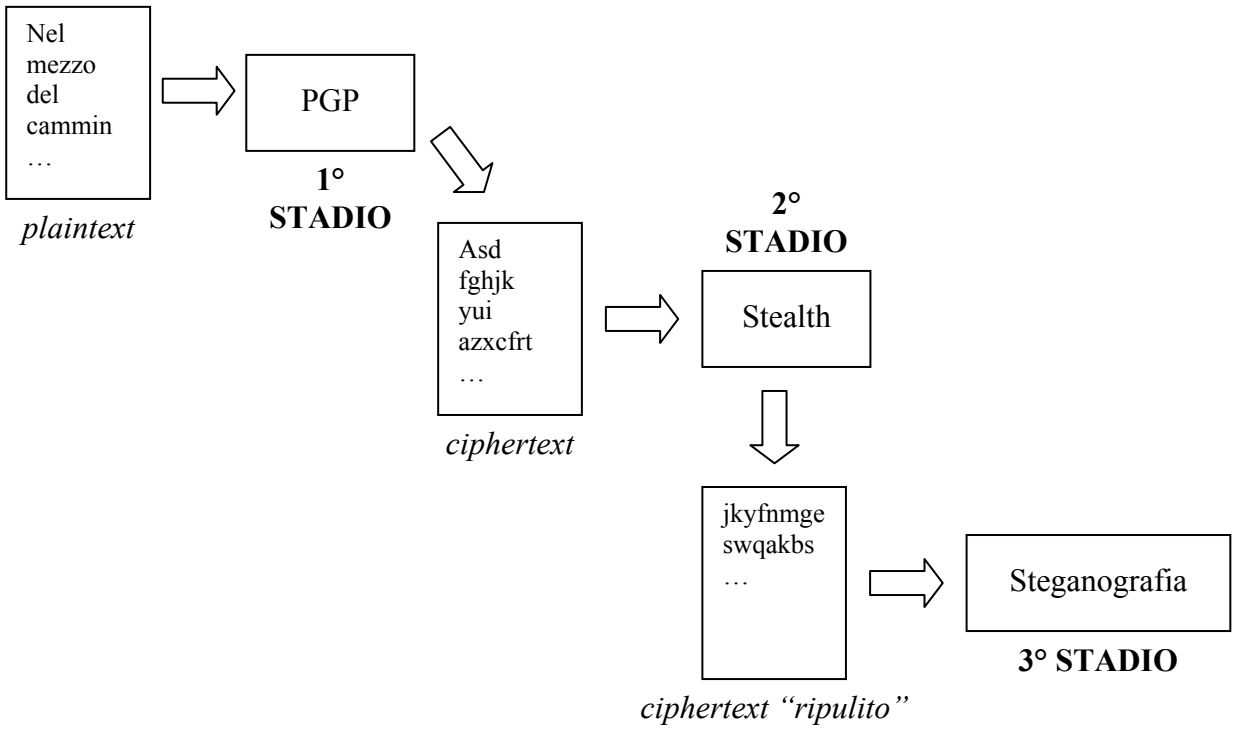


Figura V.3 – Sistema steganografico completo

L'analisi dell'iter precedente dovrebbe far nascere il seguente quesito:
"poiché è necessario, per aumentare la sicurezza del sistema, prevedere una pre-elaborazione crittografica e poi una elaborazione steganografica, non sarebbe meglio costruire direttamente dei sistemi che permettono di cifrare ed occultare il messaggio contemporaneamente? ".

Se la risposta alla domanda precedente è sì, allora S^2C è quello che stiamo cercando. Si è detto (si veda la fine del paragrafo IV.5), senza dimostrarlo, che la procedura S^2C, oltre ad occultare il messaggio segreto all'interno della cover image, è in grado, al contempo, di cifrare tale messaggio. È giunto il momento di dimostrare che, effettivamente, il processo denominato S^2C è in grado di

1. crittografare un messaggio con prestazioni identiche alle classiche metodologie di cifratura;

2. steganografare un messaggio senza alterare le principali caratteristiche statistiche dello stego medium (paragrafo IV.9), in linea con i programmi steganografici più diffusi oggi.

V.3 Dimostrazione delle proprietà di crittografia e steganografia di S^2C

Per compiere il proposito espresso appena sopra, si adotterà una metodologia di dimostrazione molto semplice:

Tesi *S^2C permette di crittografare e steganografare un messaggio m.*

Dimostrazione *dato m, dimostriamo, separatamente, che la procedura S^2C cifra m (crittografia) e che essa permette di occultare m all'interno di un cover object (steganografia).*

Le pagine a seguire contengono queste due dimostrazioni.

V.3.1 Analisi delle proprietà crittografiche di S^2C

Prima di addentrarci nel confronto tra l'algoritmo S^2C e la crittografia, forniamo una definizione precisa del *cifrario di Vernam* [22].

Definizione V.1 *Il cifrario di Vernam è definito sull'alfabeto $A = \{0, 1\}$. Un messaggio in forma binaria $m_0m_1...m_t$ è elaborato attraverso una chiave $k_0k_1...k_t$, anch'essa binaria e della stessa lunghezza, con il fine di produrre un ciphertext $c_0c_1...c_t$ dove*

$$c_i = m_i \oplus k_i, \quad 0 \leq i \leq t.$$

Il plaintext $m_0m_1...m_t$ si riottene semplicemente invertendo il procedimento

$$m_i = c_i \oplus k_i, \quad 0 \leq i \leq t.$$

Se la chiave è scelta in modo casuale e mai più riusata, il cifrario di Vernam prende il nome di one-time system oppure one-time pad *(si veda l'esempio I.1).*

Se la chiave viene riusata, è possibile attaccare il sistema. Per esempio, se $c_0c_1...c_t$ e $c_0'c_1'...c_t'$ sono due ciphertext prodotti dalla stessa chiave $k_0k_1...k_t$ allora

$$c_i = m_i \oplus k_i, \quad c_i' = m_i' \oplus k_i$$

e $c_i \oplus c_i' = m_i \oplus m_i'$. La ridondanza presente nell'ultima equivalenza può permettere di effettuare crittoanalisi.
È stato dimostrato [22] che one-time pad è teoricamente impossibile da rompere. Cioè, se un crittoanalista fosse in possesso di un ciphertext $c_0c_1...c_t$ cifrato usando una chiave casuale (random) che sia stata impiegata una sola volta, egli non potrebbe fare di meglio che cercare di indovinare la stringa di numeri binari, di lunghezza t+1, che rappresenta il plaintext: in pratica, qualsiasi stringa binaria di lunghezza t+1 ha la stessa probabilità di essere quella cercata rispetto a tutte le altre possibili. Shannon ha dimostrato (si veda il paragrafo I.1) che realizzare un sistema perfettamente sicuro richiede una chiave random della stessa lunghezza del messaggio: evidentemente questa condizione aumenta le difficoltà legate alla diffusione della chiave ed, in seconda battuta, alla sua gestione.
*A causa di queste difficoltà, è stato proposto l'impiego di chiavi (di lunghezza pari a quella del messaggio) create in maniera pseudo-random a partire da stringhe (generative) più corte, con l'intento di produrre una chiave che appaia random agli occhi di un avversario. Queste chiavi pseudo-random non offrono una sicurezza perfetta (la lunghezza delle stringhe generative è minore di quella del messaggio), ma possono essere considerate **sicure in pratica**, poichè si ipotizza (ragionevolmente) che l'avversario abbia una potenza computazionale limitata.*

Si riprenda, ora, l'esempio IV.5 esposto alla fine del capitolo precedente. Trasformiamo gli array di interi segnati lì presentati in array di numeri binari (a quattro bit) in complemento a due ottenendo la tabella subito sotto.

messaggio[ ]	*keyAC[ ]*	*stegoAC[ ]*
0	010**0**	111$\underline{0}$
1	000**1**	111$\underline{0}$
1	000**1**	111$\underline{0}$
1	111**1**	001$\underline{0}$
0	001**1**	1111
0	001**0**	111$\underline{0}$

Tabella V.1 – Confronto tra S^2C e cifrario di Vernam

Si consideri l'array messaggio[ ] come il plaintext $m_0m_1...m_t$ del cifrario di Vernam e sia l'insieme dei bit meno significativi di keyAC[ ] la chiave $k_0k_1...k_t$ (valori in grassetto in tabella V.1).
Se si costruisce il ciphertext $c_0c_1...c_t$ si ottiene l'array [0, 0, 0, 0, 1, 0], infatti:

$$0 \oplus \mathbf{0} = \underline{0}, \quad 1 \oplus \mathbf{1} = \underline{0}, \quad 1 \oplus \mathbf{1} = \underline{0}, \quad 1 \oplus \mathbf{1} = \underline{0}, \quad 0 \oplus \mathbf{1} = \underline{1}, \quad 0 \oplus \mathbf{0} = \underline{0}.$$

Se si confronta il ciphertext con l'insieme dei bit meno significativi dei valori dell'array stegoAC[ ] (sottolineati in tabella V.1) si scopre che essi sono uguali. Quindi i bit meno significativi dei coefficienti DCT (espressi in binario) conservati da stegoAC[ ] sono nient'altro che i bit del messaggio **cifrati con il metodo di Vernam**.
Se la key image fosse utilizzata una sola volta, poiché la sua dimensione è sempre almeno pari alla lunghezza del messaggio (questo è vero per costruzione, si veda il passo 3C dell'algoritmo di decodifica nel sotto paragrafo IV.9.3) si avrebbe, addirittura, una cifratura teoricamente impossibile da rompere → one-time pad.
Le considerazioni sovra esposte valgono in generale poiché la routine di codifica (passo 5C) definita nel sotto paragrafo IV.9.3 modifica l'ultimo bit di una parte (e ne lascia inalterata la restante) dei coefficienti DCT del cover medium proprio in accordo al cifrario di Vernam.

Attenzione *La procedura S^2C NON sovrascrive (overwrite) l'ultimo bit dei coefficienti DCT della cover image con il risultato dello XOR tra i bit di messaggio[ ] e i LSBs di keyAC[ ]. Infatti, l'overwriting dei bit meno significativi dei coefficienti DCT tende a rendere uguali le frequenze di occorrenza nell'istogramma dei coefficienti della immagine stego. Queste frequenze, senza modifiche steganografiche, sono sempre diseguali (si veda la figura IV.23): basterebbe, allora, un semplice test statistico per smascherare la presenza di un messaggio segreto.*
La direzione da seguire [19] è quella di sostituire l'operazione di overwriting con operazioni di incremento e decremento; in questo modo le frequenze di occorrenza dei diversi coefficienti non tenderanno ad eguagliarsi, ma piuttosto subiranno delle modifiche molto meno evidenti (le frequenze varieranno in un intervallo limitato di valori, restando comunque diseguali).

Tesi V.1 *S^2C e cifratura di Vernam sono equivalenti.*

Dimostrazione V.1 Il cifrario di Vernam è definito sull'alfabeto binario $A = \{0, 1\}$. L'algoritmo S^2C utilizza i coefficienti DCT, i quali appartengono ad un alfabeto I che comprende tutti gli interi $i \in [-2048, 2047]$ (prima della fase di quantizzazione). Vogliamo creare una relazione R che mappi l'alfabeto I sull'alfabeto A. Come è evidente, R non potrà essere biunivoca, ma al massimo univoca da I ad A; infatti la cardinalità di A è 2, di gran lunga inferiore alla cardinalità di I (che è 4096).
La più semplice ed immediata relazione R consiste nell'estrarre il bit meno significativo di ogni singolo coefficiente DCT e, in base al suo valore, R-relazionarlo a 0 oppure ad 1. Sarebbe necessario, quindi, trasformare ogni coefficiente DCT in un numero binario in complemento a due ed estrarre successivamente il LSB.
In modo più semplice, si può notare che se il coefficiente DCT è pari il suo LSB è zero, se è dispari il LSB è uno; basta, perciò, stabilire se un coefficiente è pari o dispari per sapere che il suo LSB è, rispettivamente, zero oppure uno.
È attuabile, allora, il ragionamento che segue. Per prima cosa, si trasforma l'alfabeto I in un alfabeto binario $I' = \{\text{pari, dispari}\}$ tramite una relazione univoca B; questa relazione divide l'insieme dei coefficienti DCT in due classi di appartenenza, i

coefficienti pari ed i coefficienti dispari. In altre parole, poiché un coefficiente o è pari o è dispari, i coefficienti DCT invece che sull'alfabeto I, possono essere considerati definiti su un nuovo alfabeto binario, di cardinalità pari a 2, I' = {pari, dispari} tramite una relazione B, in simboli $I \xrightarrow{B} I'$.
La mappatura R tra alfabeto I ed alfabeto A può, allora, avvenire passando attraverso la trasformazione B intermedia da I ad I'; una volta terminata questa riclassificazione (nelle due nuove classi, i pari e i dispari), tutti i coefficienti DCT apparterranno all'alfabeto I' = {pari, dispari} con pari $\xrightarrow{R}$ 0 e dispari $\xrightarrow{R}$ 1.
Mostriamo ora come far corrispondere gli elementi costituenti l'algoritmo di Vernam ai passi dell'algoritmo S^2C.
Nel cifrario di Vernam un messaggio in forma binaria $m_0m_1...m_t$ è elaborato attraverso una chiave $k_0k_1...k_t$, anch'essa binaria e della stessa lunghezza, con il fine di produrre un ciphertext $c_0c_1...c_t$ dove

$$c_i = m_i \oplus k_i, \ 0 \leq i \leq t.$$

Nella procedura S^2C, poiché il messaggio segreto è trasformato in un array di bit che prende il nome di messaggio[ ] (passo 3C della fase di codifica nel paragrafo IV.9.3), possiamo mappare, attraverso una relazione T biunivoca, messaggio[ ] su $m_0m_1...m_t$ ponendo semplicemente $m_i \xleftrightarrow{T}$ messaggio[i] con $0 \leq i \leq t$ e T relazione di uguaglianza.
Definiamo, poi, la relazione U biunivoca tra keyAC[ ] (passo 2C della fase di codifica) e chiave di Vernam $k_0k_1...k_t$:
keyAC[ ], attraverso le relazioni univoche B (che produce un array temporaneo keyPariDispari[ ] con alfabeto I') ed R, viene mappato, elemento per elemento, sull'alfabeto A, a seconda che il coefficiente DCT di keyAC[i] sia pari o dispari, producendo un array **binario** che indicheremo con key[ ].
Finita la mappatura, la relazione U altro non è se non la relazione di uguaglianza

$$\text{keyAC}[i] \xrightarrow{B} \text{keyPariDispari}[i] \xrightarrow{R} \text{key}[i] \xleftrightarrow{U} k_i \text{ con } 0 \leq i \leq t.$$

Resta da dimostrare che il passo 5C della fase di codifica (sempre paragrafo IV.9.3) è equivalente al passo di produzione del ciphertext, $c_i = m_i \oplus k_i, \ 0 \leq i \leq t$.
Il ciphertext prodotto da S^2C è costituito dall'array coverAC[ ], da modificare in base al messaggio che si vuole inserire.
Definiamo una relazione V, con lo stesso procedimento usato per U:

$$\text{coverAC}[i] \xrightarrow{B} \text{coverPariDispari}[i] \xrightarrow{R} \text{cover}[i] \xleftrightarrow{V} c_i \text{ con } 0 \leq i \leq t.$$

Si ripropone, per comodità, il passo 5C di codifica.

Passo 5C Modifica del coefficiente coverAC[i] secondo la seguente routine

if message[i] = 1 //si deve codificare un 1	linea 1
then	2
if coverAC[i] e keyAC[i] sono entrambi pari od entrambi dispari	3

```
        then  incrementa o decrementa coverAC[i]                    linea  4
        end if                                                              5
        else  non fare nulla                                                6
        end else                                                            7
    end if                                                                  8
    else  //si tratta del caso message[i] = 0, si deve codificare uno 0     9
        if  coverAC[i] e keyAC[i] sono uno pari ed uno dispari              10
        then  incrementa o decrementa coverAC[i]                            11
        end if                                                              12
        else  non fare nulla                                                13
        end else                                                            14
end else                                                                    15
```

Modifichiamo la routine di sopra ponendo key[ ] al posto di keyAC[ ], cover[ ] al posto di coverAC[ ] e $m_0m_1...m_t$ al posto di message[ ]. Le linee 3 e 4 di 5C vengono modificate inserendo l'operatore XOR, che permette di rappresentare (applicandosi a numeri binari) il confronto tra coefficienti pari e dispari (linea 3) e la trasformazione da pari a dispari e viceversa (linea 4); la stessa cosa dicasi per le linee 10 e 11.

Passo 5C modificato (in seguito indicato con **5M**)

```
    if  m_i = 1 //si deve codificare un 1                               linea  1
    then                                                                       2
        if  cover[i]  ⊕ key[i] = 0                                             3
        then  cover[i] = cover[i]  ⊕ 1                                         4
        end if                                                                 5
        else  non fare nulla                                                   6
        end else                                                               7
    end if                                                                     8
    else  //si tratta del caso m_i = 0, si deve codificare uno 0               9
        if  cover[i]  ⊕ key[i] = 1                                             10
        then  cover[i] = cover[i]  ⊕ 1                                         11
        end if                                                                 12
        else  non fare nulla                                                   13
        end else                                                               14
end else                                                                       15
```

Mostriamo la tavola di verità per $c_i = m_i \oplus k_i$.

m_i	k_i	c_i
0	*0*	*0*
0	*1*	*1*
1	*0*	*1*
1	*1*	*0*

Tabella V.2 – Tavola di verità per $c_i = m_i \oplus k_i$

Esibiamo la tavola di verità per per $m_i = c_i \oplus k_i$.

c_i	k_i	m_i
0	*0*	*0*
0	*1*	*1*
1	*0*	*1*
1	*1*	*0*

Tabella V.3 – Tavola di verità per $m_i = c_i \oplus k_i$

Facciamo vedere che il passo 5M rispetta la tabella V.2 che rappresenta la fase di codifica. La decifratura (in tabella V.3) è simmetrica alla precedente.
Dato m_i = 1, bisogna che cover[*i*] e key[*i*] siano uno 0 e l'altro 1, altrimenti non sarebbe possibile recuperare il valore 1 in decodifica (terza e quarta riga della tabella V.2). Se cover[*i*] e key[*i*] sono entrambi 0 o entrambi 1 (linea 3 del passo 5M), cover[*i*] deve diventare 0 se era 1 e viceversa. Questo lo si fa negando cover[*i*] (linea 4 del passo 5M). Se, al contrario, cover[*i*] e key[*i*] sono uno 0 e l'altro 1 non bisogna far nulla (linea 6 di 5M). Applicando 5M, per m_i = 1, si ottiene la tabella V.4.

message[i]	*key[i]*	*cover[i]*	*key[i] ⊕ cover[i]*	*cover[i] dopo la modifica*
1	0	0	0	1
1	0	1	1	1
1	1	0	1	0
1	1	1	0	0

Tabella V.4 – Risultati del passo 5M, per message[i] = 1

Le righe delle colonne prima (*message[i]*), seconda (*key[i]*) e quinta (*cover[i] dopo la modifica*) della tabella V.4 sono perfettamente uguali (salvo ripetizioni) alle righe tre e quattro della tabella V.2, permettendo di affermare che, nel caso m_i = 1, il passo 5M e l'algoritmo di Vernam sono equivalenti.
Ragionamento analogo (che il lettore potrà facilmente verificare) vale per il secondo caso m_i = 0, portando alla tabella V.5.

message[i]	*key[i]*	*cover[i]*	*key[i] ⊕ cover[i]*	*cover[i] dopo la modifica*
0	0	0	0	0
0	0	1	1	0
0	1	0	1	1
0	1	1	0	1

Tabella V.5 – Risultati del passo 5M, per message[i] = 0

Di nuovo, le righe delle colonne prima, seconda e quinta della tabella V.5 sono uguali (salvo ripetizioni) alle righe uno e due della tabella V.2, permettendo di affermare

che, anche nel caso $m_i = 0$, il passo 5M e l'algoritmo di Vernam sono equivalenti. Concludiamo, allora, che passo 5M e algoritmo di Vernam sono equivalenti in ogni caso (cioè sia se $m_i = 0$ sia se $m_i = 1$).
Poiché il passo 5M realizza la cifratura di Vernam (lo abbiamo appena visto) e poiché il passo 5M è stato ottenuto dal passo 5C di codifica della procedura S^2C, semplicemente applicando le trasformazioni di uguaglianza T, U e V, il passo 5C di codifica in S^2C realizza la cifratura di Vernam. L'unica differenza tra la codifica di S^2C e la cifratura di Vernam è che la prima lavora su cifre pari e dispari (alfabeto I'), la seconda su cifre binarie (alfabeto A); giacché si possono mappare tra loro biunivocamente gli insiemi I' ed A, S^2C e cifrario di Vernam sono equivalenti in fase di codifica.
Il processo di decodifica è simmetrico rispetto alla codifica, per cui si tratta solo di ripetere con la routine 5D le considerazione già fatte per 5C (si risparmia al lettore questa ulteriore prova).
In virtù della trattazione appena presentata, si afferma, senza possibilità di essere smentiti, che algoritmo S^2C e cifratura di Vernam sono equivalenti. *c.v.d.*

Abbiamo dimostrato che la procedura S^2C modifica i coefficienti della cover image producendo la stego image in modo tale che quest'ultima rappresenti il ciphertext relativo al messaggio segreto.
Mostriamo ora che S^2C è un sistema di steganografia in grado di conservare inalterate le principali caratteristiche statistiche dello stego file.

V.3.2 Analisi delle proprietà steganografiche di S^2C

Ipotesi: "*S^2C è un sistema di steganografia che preserva le qualità statistiche della stego image*".
Dimostriamolo attraverso un confronto comparativo con uno dei più diffusi software steganografici, F5 di Pfitzmann e Westfeld [35], già descritto nel paragrafo III.10.
È molto importante notare che F5, anche se modifica l'istogramma dei coefficienti DCT, conserva le due caratteristiche fondamentali dell'istogramma JPEG discusse nel paragrafo IV.9.1. Inoltre, F5 non va soggetto né all'attacco χ^2 nè all'attacco visuale. Le dimostrazioni di queste proprietà sono state fornite, in maniera formale, da Pfitzmann e Westfeld in [35].
Qui dimostriamo, in ordine, che la metodologia S^2C non è soggetta agli attacchi visuali e, per confronto diretto con F5, resiste agli attacchi statistici.

Tesi V.2 *S^2C non è soggetto agli attacchi visuali perché adopera immagini JPEG come cover.*

Dimostrazione V.2 *Gli attacchi visuali (paragrafo III.8) ai sistemi steganografici sono basati sull'analisi delle strutture visuali presenti nel cover medium che vengono, in misura consistente, sovrascritte dall'algoritmo steganografico. È stato dimostrato [19] che l'overwriting del LSB dei valori di colore di una immagine*

provoca artefatti individuabili nello stego medium, poiché è possibile vedere la struttura dell'immagine anche sul piano immagine costituito dai bit meno significativi (questo accade usualmente nelle immagini BMP). Se si va a modificare questo piano, le strutture presenti nell'immagine "normale" (cioè, con tutti i piani) scompaiono nell'immagine ad un unico piano (si veda l'esempio III.4).
Le tecniche adattive, che portano il rate di inserimento in linea con il contenuto del cover, prevengono gli attacchi visuali; tuttavia esse riducono notevolmente la percentuale di informazione steganografica inseribile nel cover rispetto alle dimensioni di quest'ultimo. Dal momento che i cover object compressi con formati lossy (JPEG e mp3 giusto per citarne due) sono di per se stessi adattivi, essi non subiscono attacchi visuali. Conclusione: la procedura S^2C sfrutta immagini JPEG quindi è immune da attacchi visuali. c.v.d.

Confrontiamo le performance dell'algoritmo S^2C con quelle di F5, adoperando lo stesso file messaggio, la stessa immagine cover e lo stesso indice di qualità JPEG.

Esempio V.1 *Sia m (riportato in basso) il messaggio segreto che si vuole nascondere all'interno della cover image C (figura V.4).*

m: Steganography is the art and science of communicating in a way which hides the existence of the communication. In contrast to cryptography, where the "enemy" is allowed to detect, intercept and modify messages without being able to violate certain security premises guaranteed by a cryptosystem, the goal of steganography is to hide messages inside other "harmless" messages in a way that does not allow any "enemy" to even detect that there is a second secret message present [Markus Kuhn 1995-07-03]. *

* questo testo è usato come test in [6].

C ha dimensioni 640 × 480, è in formato JPEG e pesa 30,7 KB :

Figura V.4 – La cover image C

Il nostro proposito è quello di utilizzate m e C come input per l'algoritmo S^2C e per F5, per poi confrontare i due istogrammi delle frequenze di occorrenza dei vari coefficienti DCT estratti dalle stego image prodotte dai due metodi. Definiamo, a tal proposito, stego image S l'output di F5 ed S' l'output della procedura S^2C.
Usiamo prima F5 con i seguenti parametri:

- *fattore di qualità Q della stego image = 75,*
- *file di input (TIFF, BMP, JPEG o GIF) = C (JPEG),*
- *il nome del file output = S,*
- *il file contenente il messaggio segreto = m,*
- *una user password da usare come seme per il generatore di numeri pseudo random (PRNG) = "provaF5",*
- *il commento da inserire nell'header del file JPEG = "confronto".*

L'output S di F5 è riportato in figura V.5. S ha le stesse dimensioni di C ed un peso di 30,1 KB.

Figura V.5 – Stego image S prodotta da F5

L'algoritmo S^2C ha bisogno di un ulteriore input, la key image che indicheremo con la lettera maiuscola K.
K (figura V.6) ha dimensioni 210 × 160, è in formato JPEG e pesa 6.86 KB :

Figura V.6 – La key image K

Osserviamo ora la stego image S' prodotta adoperando S^2C con indice di qualità sempre 75 (figura V.7). S' ha le stesse dimensioni di C ed un peso di 30,7 KB.

Figura V.7 – Stego image S'

Il confronto visivo tra le due immagini non permette di individuare differenze. Infatti, l'occhio umano è incapace di rilevare le sottili dissomiglianze tra C, S ed S'.
È necessario, allora, analizzare le proprietà statistiche delle tre immagini di cui sopra. Il grafico V.1 riporta l'istogramma ai coefficienti AC dell'immagine C. Per aumentare la leggibilità non è stata inserita la colonna relativa allo zero che presenta valori enormemente superiori ai valori di frequenza degli altri; in ogni caso, tutti i valori in dettaglio sono riportati nella tabella V.6. Il nostro obiettivo sarà quello di confrontare i valori riportati nel grafico V.1, relativi a C, con i valori presentati da S e da S' (grafici V.2 e V.3). Se S' presenta, nei confronti di C, le stesse differenze percentuali che evidenzia S rispetto a C, allora S^2C resiste agli attacchi statistici perché è noto che F5 vi resiste.
I grafici V.2 e V.3 nelle pagine seguenti sono, in ordine, gli istogrammi per S e per S'. La loro presenza è utile per dimostrare qualitativamente che sia F5 sia S^2C preservano le due caratteristiche proprie di ogni immagine JPEG "naturale", cioè non alterata da contenuti steganografici:

1. *la frequenza di occorrenza dei coefficienti decresce all'aumentare del loro valore assoluto;*
2. *la diminuzione della frequenza di occorrenza dei coefficienti decresce all'aumentare del loro valore assoluto, cioè la differenza tra due barre dell'istogramma nel mezzo del grafico è maggiore rispetto a quella che si riscontra ai lati.*

Il lettore può verificare, per ispezione visiva, la fondatezza di tale affermazione.

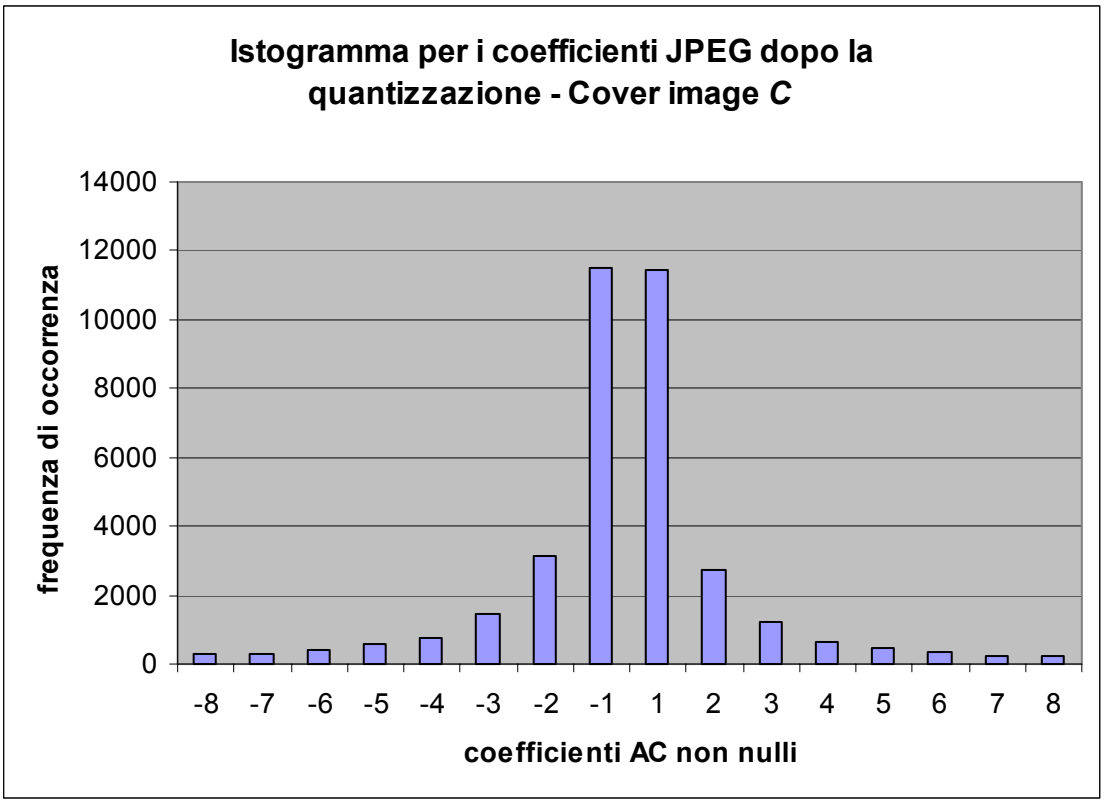


Grafico V.1 – Istogramma per C

coefficienti	*frequenza*
0	417308
+1	11432
-1	11499
+2	2718
-2	3149
+3	1221
-3	1424
+4	667
-4	762
+5	470
-5	560
+6	345
-6	397
+7	238
-7	315
+8	222
-8	280

Tabella V.6 – Frequenze dei coefficienti AC di C

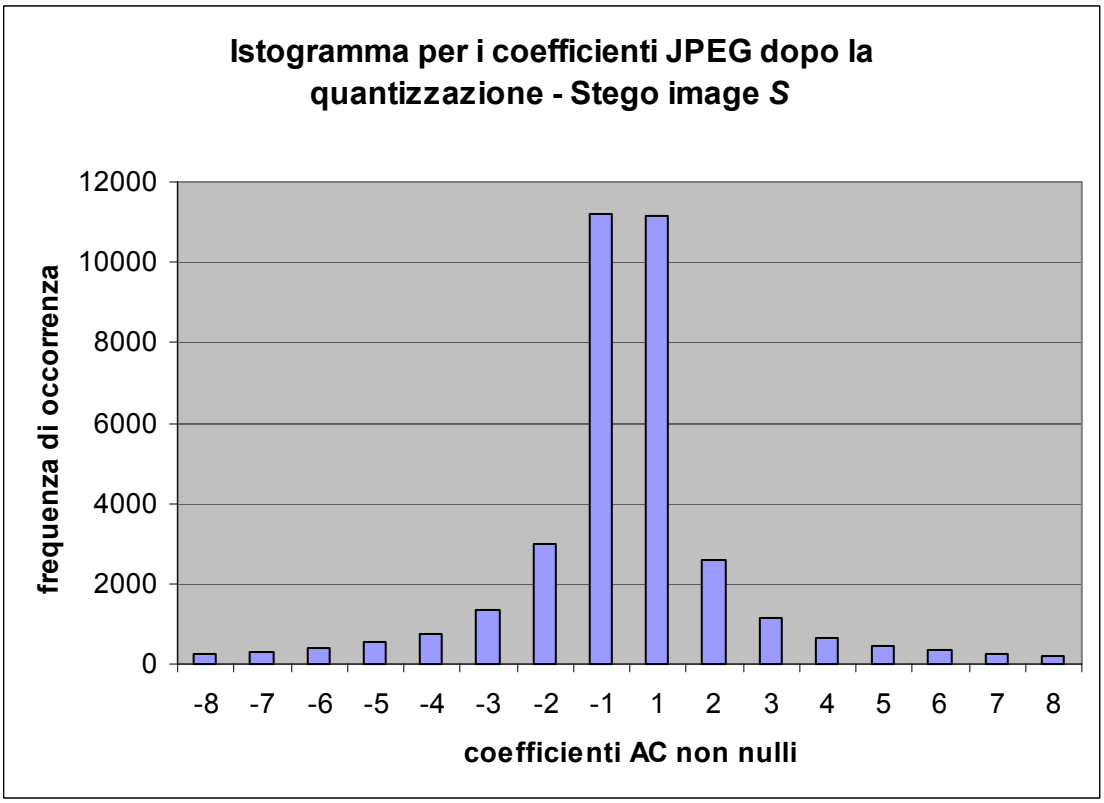


Grafico V.2 – Istogramma per S

coefficienti	*frequenza*	*differenza S, C*	*differenza in %*
0	418399	1091	0.26 %
+1	11151	281	2.45 %
-1	11218	281	2.44 %
+2	2566	152	5.59 %
-2	2997	152	4.82 %
+3	1162	59	4.83 %
-3	1346	78	5.47 %
+4	655	12	1.80 %
-4	765	3	0.39 %
+5	462	8	1.70 %
-5	538	22	3.93 %
+6	337	8	2.32 %
-6	389	8	2.01 %
+7	232	6	2.52 %
-7	319	4	1.27 %
+8	219	3	1.35 %
-8	269	11	3.93 %

Tabella V.7 – Frequenze dei coefficienti AC di S

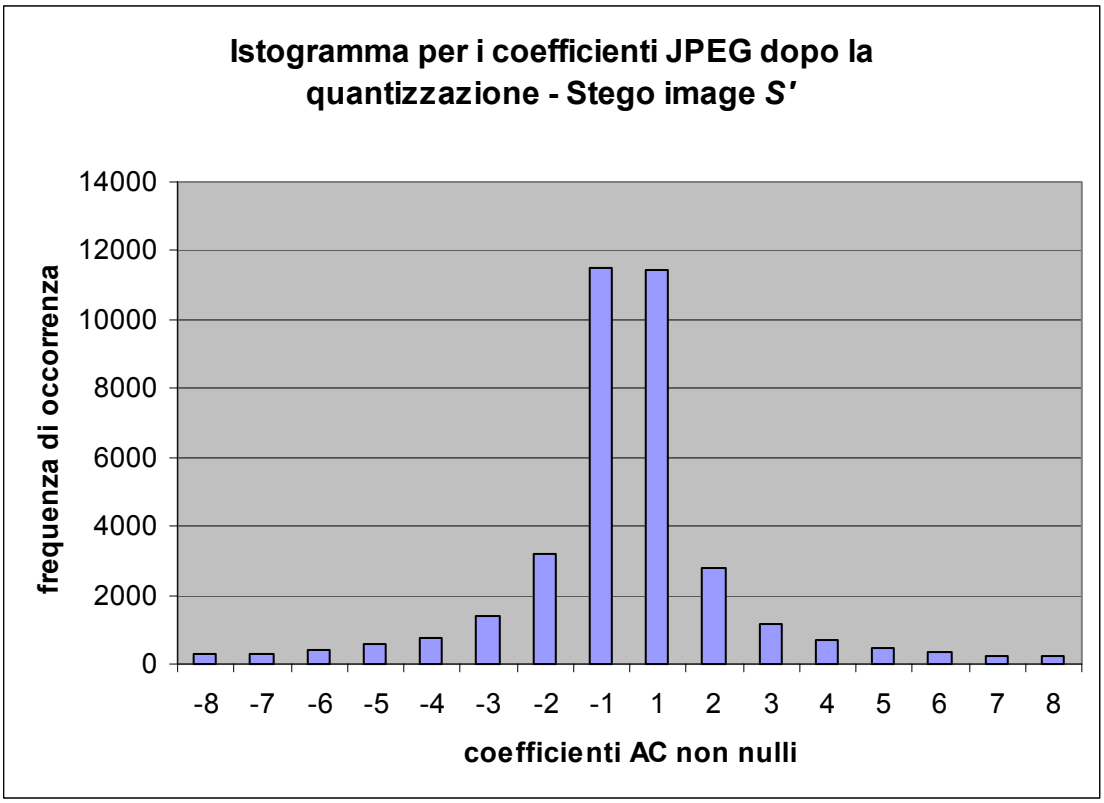


Grafico V.3 – Istogramma per S'

coefficienti	*frequenza*	*differenza S', C*	*differenza in %*
0	417308	0	0 %
+1	11431	1	0.00 %
-1	11500	1	0.00 %
+2	2743	25	0.92 %
-2	3190	41	1.30 %
+3	1183	38	3.11 %
-3	1350	74	5.19 %
+4	672	5	0.75 %
-4	781	19	2.49 %
+5	470	0	0 %
-5	569	9	1.61 %
+6	361	16	4,64 %
-6	402	5	1,26 %
+7	226	12	5.04 %
-7	313	2	0.63 %
+8	226	4	1.80 %
-8	282	2	0.71 %

Tabella V.8 – Frequenze dei coefficienti AC di S'

L'analisi dei valori riportati nelle tre tabelle V.6, V.7 e V.8 porta al grafico V.4 che permette di realizzare un efficace confronto diretto tra S ed S'.

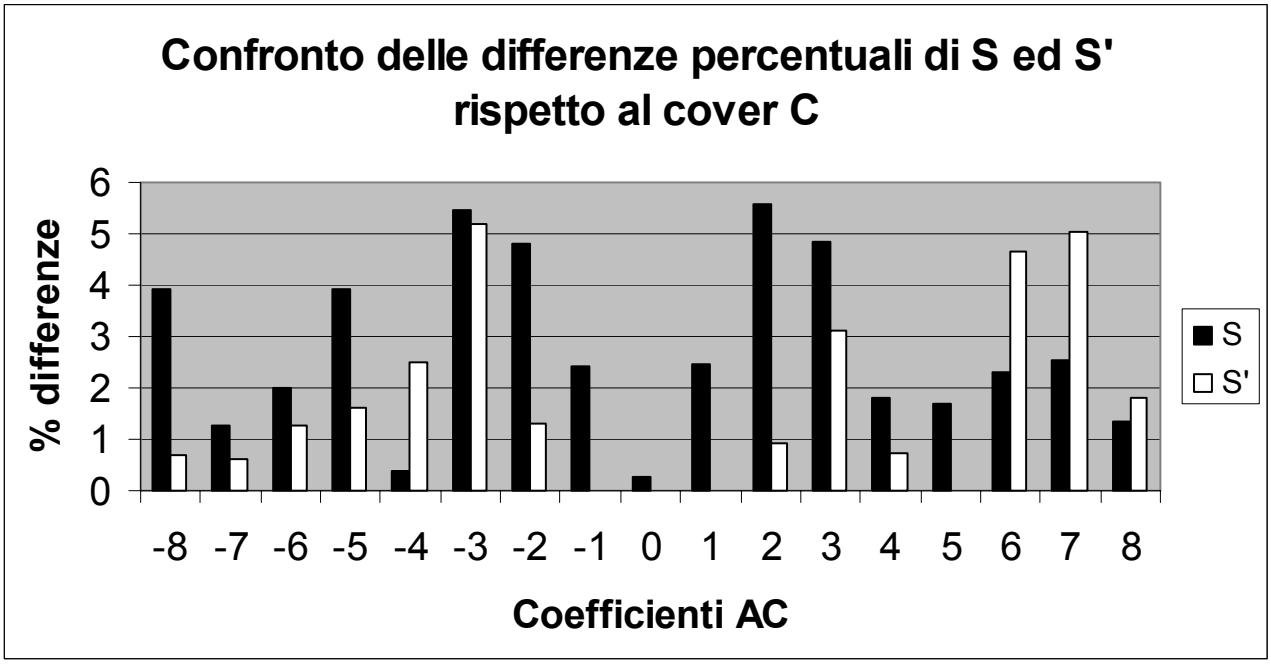


Grafico V.4 – Confronto delle differenze % tra S ed S'

Le differenze percentuali per F5 e per S^2C sono dello stesso ordine di grandezza, comunque contenute ben all'interno del 6% per entrambi gli algoritmi. Si fa notare che S^2C non inserisce differenze sullo zero perché non considera i coefficienti nulli nella fase di embedding del messaggio, mentre F5 fa aumentare il numero di coefficienti zero quando è costretto a modificare i coefficienti pari ad 1 e -1 (cioè quando si verifica il già citato shrinkage). Inoltre, S^2C possiede un meccanismo di correzione statistica per il riallineamento dei coefficienti pari ad 1 e -1, che azzera la differenza percentuale su quei coefficienti. Va precisato che suddetto automatismo può essere utilizzato solo se il messaggio segreto non satura completamente la capacità del cover medium (si rimanda al paragrafo V.4 per la trattazione dell'algoritmo di correzione di S^2C e per la quantificazione della capacità massima di informazione steganografica che può essere trasportata dalla cover image).
Sommando tutte le differenze percentuali sulla frequenza dei coefficienti nell'intervallo [-8, 8] riportate nel grafico V.4 si ottiene:

$$\text{somma delle differenze percentuali per } S^2C = 29{,}45$$

$$\text{somma delle differenze percentuali per F5} = 47{,}08$$

Il confronto tra i due valori appena ottenuti ci permette di affermare che S^2C ha un comportamento equiparabile, per quanto riguarda la conservazione delle frequenze di occorrenza dei coefficienti AC, a quello di F5. □

L'esempio V.1 ha permesso di mostrare come non vi siano sostanziali discrepanze nelle differenze percentuali introdotte da F5 rispetto a quelle introdotte da S^2C.

Per rafforzare tale tesi, esponiamo un nuovo esempio.

Esempio V.2 *Sia m (riportato in basso) il messaggio segreto che si vuole nascondere all'interno della cover image C (figura V.8).*

m: Non è questo il racconto di gesta impressionanti, ma neppure quel che si direbbe semplicemente "un racconto un po' cinico"; per lo meno, non vuole esserlo. E' un segmento di due vite raccontate nel momento in cui hanno percorso insieme un determinato tratto, con la stessa identità di aspirazioni e sogni. Un uomo nell'arco di nove mesi della sua vita può pensare a molte cose, dalla più alta speculazione filosofica sino al più basso anelito per un piatto di minestra, in totale correlazione con lo stato di vacuità del suo stomaco; e se al tempo stesso ha in sé qualcosa dell'avventuriero, in questo lasso di tempo può vivere momenti che forse risulteranno interessanti ad altre persone, e il cui racconto spassionato risulterebbe qualcosa di simile a questi appunti. *

* questo testo è tratto da "Latinoamericana, diario di un viaggio in motocicletta", di Ernesto Guevara de la Serna, edizioni Feltrinelli.

C ha dimensioni 800 × 600, è in formato JPEG e pesa 420 KB :

Figura V.8 – La cover image C

Come prima, utilizzeremo m e C come input per l'algoritmo S^2C e per F5, per poi confrontare i due istogrammi delle frequenze di occorrenza dei vari coefficienti DCT estratti dalle stego image prodotte dai due metodi. Definiamo, a tal proposito, stego image S *l'output di F5 ed* S' *l'output della procedura S^2C.*
Usiamo prima F5, con i seguenti parametri:

- *fattore di qualità Q della stego image = 75,*
- *file di input (TIFF, BMP, JPEG o GIF) = C (JPEG),*
- *il nome del file output = S,*

- *il file contenente il messaggio segreto = m,*
- *una user password da usare come seme per il generatore di numeri pseudo random (PRNG) = "provaF5",*
- *il commento da inserire nell'header del file JPEG = "confronto".*

L'output S di F5 è riportato in figura V.9. S ha le stesse dimensioni di C ed un peso di 52,5 KB.

Figura V.9 – Stego image S prodotta da F5

L'algoritmo S^2C ha bisogno di un ulteriore input, la key image che indicheremo con la lettera maiuscola K.
K (figura V.10) ha dimensioni 800 × 600, è in formato JPEG e pesa 144 KB :

Figura V.10 – La key image K

Osserviamo ora (figura V.11 a pagina seguente) la stego image S' prodotta adoperando S^2C, con indice di qualità sempre 75. S' ha le stesse dimensioni di C ed un peso di 53,4 KB.

Figura V.7 – Stego image S'

L'analisi dei valori di frequenza presentati dalle immagini di cui sopra, omessi per non appesantire l'esempio, permette di realizzare un confronto diretto tra S ed S'.

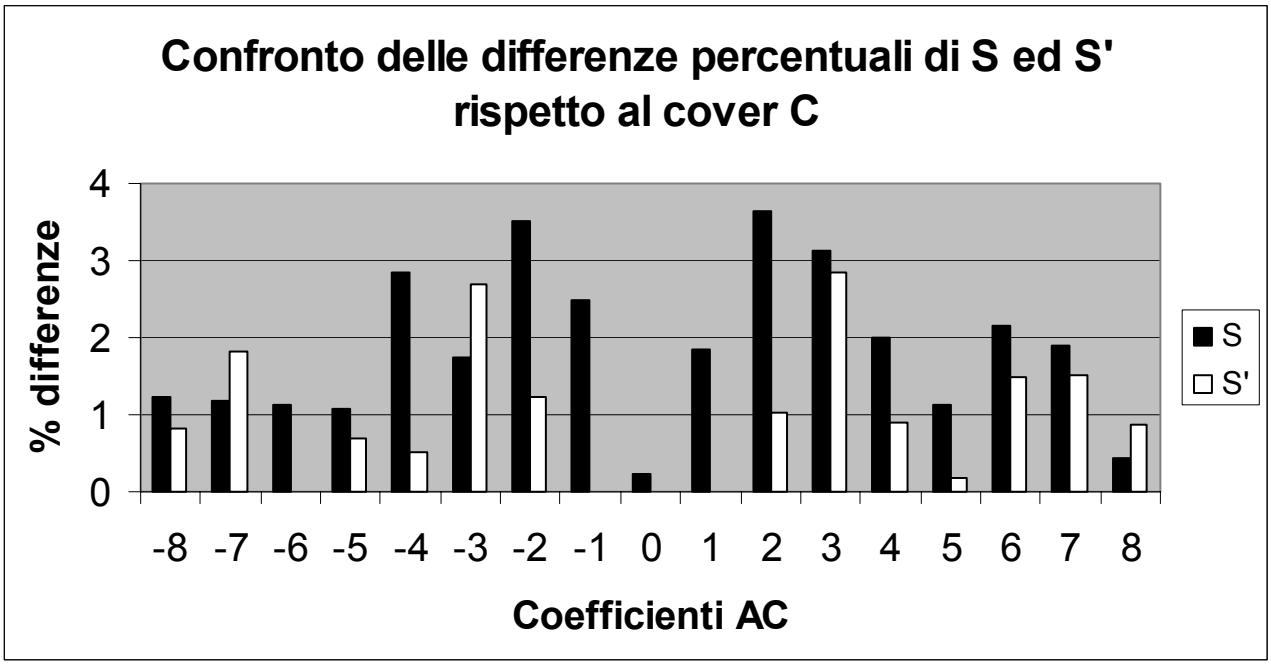


Grafico V.5 – Confronto delle differenze % tra S ed S'

Come per l'esempio precedente, le differenze in percentuale per F5 e per S^2C sono dello stesso ordine di grandezza, tutte sotto il 4%.
Sommando tutte le differenze percentuali rispetto alla frequenza dei coefficienti nell'intervallo [-8, 8] riportate nel grafico V.5 si ottiene:

somma delle differenze percentuali per S^2C = 16,57

somma delle differenze percentuali per F5 = 25,68

Il confronto tra i due valori appena ottenuti ci permette, ancora, di affermare che S^2C ha un comportamento equiparabile, per quanto riguarda la conservazione delle frequenze di occorrenza dei coefficienti AC, a quello di F5. □

Tesi V.3 *S^2C non è soggetto agli attacchi statistici perché ha un comportamento simile ad F5.*

Dimostrazione V.3 *La dimostrazione è per confronto. Dal momento che F5 resiste agli attacchi statistici dato che modifica l'istogramma delle frequenze in modo non rilevabile dall'attacco χ^2, se S^2C produce un istogramma simile ad F5 allora anche S^2C resiste agli attacchi statistici (in particolare, al test χ^2).*
Gli esempi V.1 e V.2 permettono di sostenere che S^2C produce una stego image il cui istogramma è addirittura più corrispondente, in base alle minori differenze percentuali, all'istogramma della cover image rispetto a quello prodotto da F5.
Si conclude, allora, che S^2C resiste agli attacchi statistici perché si comporta meglio di F5. *c.v.d.*

V.4 Correzione statistica in S^2C

L'algoritmo S^2C non utilizza lo zero per inserire valori steganografici, per questo se si rende necessario modificare un coefficiente pari ad 1 o -1, esso può solo diventare 2 o -2, rispettivamente. Poiché il numero di coefficienti 1 è molto maggiore del numero di coefficienti 2, se si assume che, più o meno, si debba modificare un coefficiente ogni due, il numero totale di coefficienti di valore 2 presenti nella stego image è destinato a crescere rispetto a quello che precedentemente era contenuto nella cover image; al tempo stesso va a decrescere il numero di occorrenze per gli 1. Uguale discorso vale per i coefficienti -1 e -2.

Esempio V.3 *Sia 10000 il numero di coefficienti AC pari ad 1 presenti in una immagine JPEG e sia 2000 il numero di coefficienti pari a 2. Se si assume che sia necessario modificare la stessa percentuale, 20%, di coefficienti uguali ad 1 e coefficienti corrispondenti a 2, si otterrebbe:*

- *20% di 10000 = 2000, 20% di 2000 = 400*
- *nuovo numero di coefficienti pari ad 1 = 10000 – 2000 + 400 = 8400*
- *nuovo numero di coefficienti pari a 2 = 2000 – 400 + 2000 = 3600*

È evidente che la differenza tra le due quantità tende ad assottigliarsi, fornendo la possibilità di un attacco statistico. □

S^2C per evitare l'avvicinamento delle frequenze tra numero totale di coefficienti pari ad 1 ed a 2 (-1 e -2), cerca di distribuire uniformemente il messaggio su tutta l'immagine cover, correggendo appena possibile i coefficienti inutilizzati per riallineare l'istogramma delle frequenze della stego image a quello del cover. L'esempio V.4, a pagina seguente, cerca di spiegare questo concetto.

Esempio V.4 *Sia 22000 il numero di coefficienti AC diversi da zero presenti in una immagine JPEG. Sia 3000 la lunghezza, in bit, del messaggio segreto. S^2C "spalma" il messaggio sull'immagine, inserendo un valore steganografico ogni 7 coefficienti AC, dove 7 si ottiene semplicemente calcolando il quoziente tra 22000 e 3000.*
I 6 coefficienti inutilizzati tra ogni bit del messaggio vengono impiegati per il mantenimento dell'istogramma delle frequenze originario: se è stato sostituito un 1 con un 2 (si tratta del caso più probabile), la routine di correzione di S^2C cerca un 2 tra i 6 coefficienti seguenti (che non hanno valore steganografico), se è presente lo sostituisce con un 1. □

La correzione statistica di S^2C è on-line e non a posteriori, il tutto per aumentare la velocità di esecuzione. La complessità di tale routine è lineare, poiché dipende dalla lunghezza in bit del messaggio, che poi decide il numero di coefficienti inutilizzati da esaminare.
Si fa notare che se la capacità del cover (pari al numero di coefficienti AC non nulli) viene sfruttata interamente, non è possibile effettuare una correzione statistica, poiché non sono presenti coefficienti inutilizzati da utilizzarsi per la correzione. È buona norma, perciò, utilizzare cover che presentino una capacità decisamente superiore rispetto alla lunghezza del messaggio.

V.5 One-time pad ed S^2C

Il più grosso inconveniente di one-time pad risiede nell'obbligo di utilizzare una chiave lunga quanto il messaggio, chiave impiegabile una sola volta. Tutto ciò riduce la praticabilità di one-time a pochi, e molto particolari, compiti (si veda l'esempio V.5).

Esempio V.5 *Si racconta che fino a pochi anni fa, la linea di comunicazione tra Mosca e Washington fosse cifrata per mezzo di one-time pad. Il trasporto delle chiavi era assicurato da corrieri fidati.* □

Se due persone "normali", che non siano cioè il presidente americano e quello russo, vogliono scambiarsi un messaggio confidenziale, è poco probabile che esse si inviino tra loro lunghe tavole composte solo di uni e zeri (le chiavi), per mezzo di un corriere fidato. Primo perché costerebbe troppo, secondo perché la comunicazione diverrebbe lentissima.
Escludendo i corrieri, se i due volessero scambiarsi le chiavi tramite Internet, la loro comunicazione desterebbe comunque sospetti, sempre perché nessuno, senza un secondo fine, invierebbe lunghe tavole binarie attraverso la posta elettronica.
Inoltre, sarebbe molto facile, per una entità preposta al controllo, costruire un filtro per bloccare comunicazioni che presentano una struttura così rara e particolare.
Ancora, le tavole con le chiavi non potrebbero essere inviate in chiaro, ma dovrebbero essere cifrate, generando un ciclo infinito.

L'algoritmo one-time pad classico sembra davvero essere impraticabile per scopi non eccezionali. Come fare allora?
Spesso, si è soliti inviare immagini in allegato alla posta elettronica o visitare siti che contengono immagini: non c'è nulla di strano nel fare queste due operazioni, che sono all'ordine del giorno.
A cosa porta questo discorso: se, invece di inviare una chiave binaria, inviamo una chiave che appare in tutto e per tutto un'immagine (la key image), difficilmente qualcuno potrà sospettare della nostra trasmissione. Infatti, di immagini in Internet ve ne sono miliardi ed è impossibile tenerne sotto controllo il flusso.
Ecco che S^2C usa, al posto delle chiavi binarie, chiavi che possiamo definire, giusto per intenderci, fotografiche. In altre parole, **S^2C è un one-time pad fotografico** anziché binario. L'uso delle immagini permette di abbassare di molto la probabilità che la comunicazione riservata sia bloccata perché sospetta. In più, dal momento che S^2C è progettato anche come algoritmo steganografico, deve attenersi a delle precise norme per non risultare sospetto (si deve difendere dagli attacchi steganalitici), quindi risulta molto molto difficile per un avversario bloccare la trasmissione.
Ritornando all'esempio V.5, S^2C fa in modo che il nostro corriere "fidato" sia Internet, e che le nostre tavole binarie siano semplici immagini.

Esempio V.6 *I soliti Alice e Bob vogliono scambiarsi un messaggio segreto, senza che Wendy, ancora lei, possa sospettare del loro intento. Alice e Bob condividono una password p, che Wendy non conosce.*
Alice invia a Bob una prima immagine S_1 senza alcun contenuto occulto: Alice e Bob posseggono ora S_1. Entrambi modificano S_1 inserendovi all'interno un messaggio ottenuto da p, per mezzo di un generatore di numeri pseudo-random, procurandosi una nuova immagine, che risulterà identica per entrambi, K_1; quest'ultima è la key image corrente.
Alice invia a Bob S_2, che è una immagine contenente un messaggio segreto m_1, inserito adoperando S^2C con input una immagine C_1 (scelta da Alice), m_1 e K_1. Bob riceve S_2 ed estrae m_1 grazie a K_1.
Se Alice vuole inviare un nuovo messaggio a Bob, può sfruttare l'immagine S_2 già spedita in precedenza e che Bob possiede. Come prima, Alice e Bob modificano S_2 inserendovi all'interno un messaggio ottenuto da p, per mezzo di un generatore di numeri pseudo-random, procurandosi una nuova immagine K_2, ancora identica per entrambi; K_2 è la nuova key image corrente.
Alice invia a Bob S_3 che è una immagine contenente un messaggio segreto m_2, inserito adoperando S^2C con input una immagine C_2 scelta da Alice, m_2 e K_2. Bob riceve S_3 ed estrae m_2 grazie a K_2.
Il giochetto, o per essere più precisi il protocollo, appena descritto (e riassunto in figura V.10) si può ripetere all'infinito. □

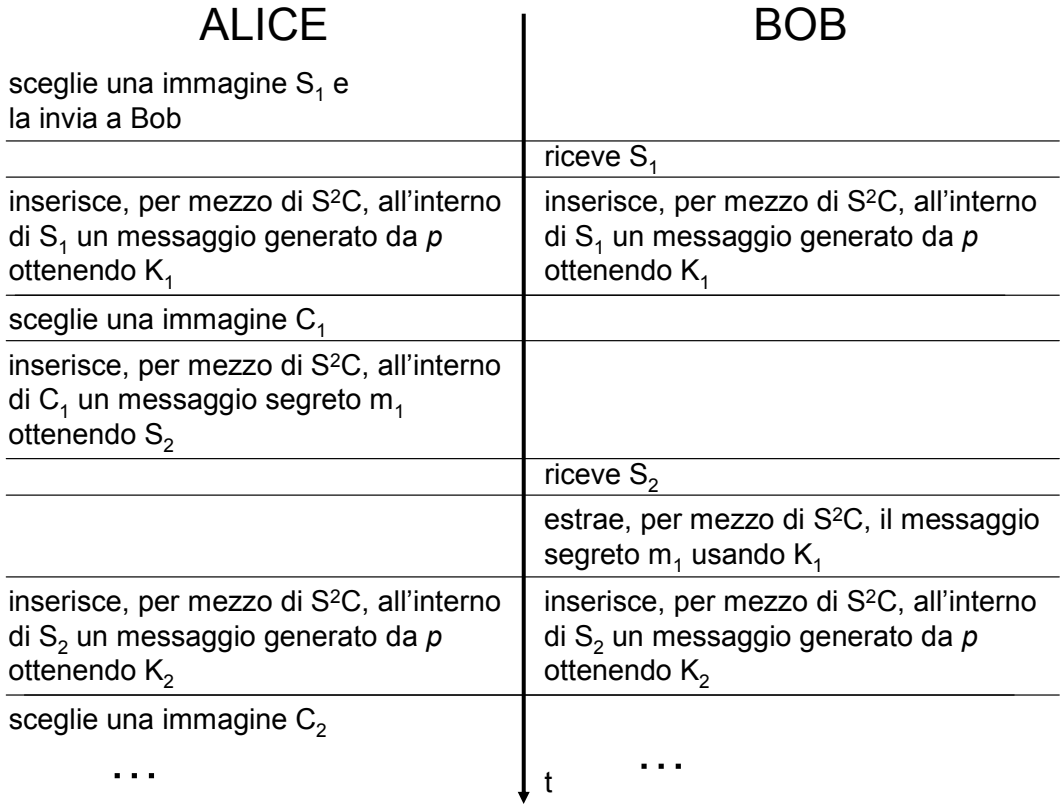


Figura V.10 – Protocollo di comunicazione per S^2C

Riprendiamo l'enunciato del principio di Kerckhoffs che era stato espresso nel paragrafo V.1 in ambito steganografico.

Principio di Kerckhoffs in ambito steganografico *la sicurezza del sistema deve basarsi sull'ipotesi che il nemico abbia piena conoscenza dei dettagli di progetto e di implementazione del sistema stesso; la sola informazione di cui il nemico non può disporre è una sequenza (corta) di numeri casuali - la chiave segreta - senza la quale, osservando un canale di comunicazione, egli non deve avere neanche la più piccola possibilità di verificare che sia in corso una comunicazione nascosta.*

La password *p*, introdotta nell'esempio V.6, può essere interpretata come la "*sequenza (corta) di numeri casuali*" richiesta nella definizione appena sopra; *p*, a differenza della key image (che non è certamente corta, in quanto deve essere lunga almeno quanto il messaggio, come richiesto dal cifrario di Vernam), può essere una parola od una frase scelta arbitrariamente. Definiamo, allora, con maggior rigore, come *p* entri a far parte del sistema S^2C.

Definizione V.1 *Si definisce* chiave utente *(o user key) una sequenza segreta e corta, di numeri o di lettere, che sender e receiver condividono e che non inviano mai sul canale di comunicazione pubblico che essi utilizzano.*

Grazie alla definizione del protocollo definito nell'esempio V.6 e nella figura V.10, si può affermare che S^2C rispetta il principio di Kerckhoffs espresso in ambito steganografico. Infatti, si è dimostrato che S^2C è resistente agli attacchi visuali e statistici, fatto che rende complicato, per il nemico, etichettare come sospetta una immagine prodotta da S^2C.
Otteniamo, infine, il seguente risultato: S^2C è un sistema steganografico valido a tutti gli effetti.

V.6 S^2C e desiderata di Kerckhoffs

Riproponiamo i desiderata di Kerckhoffs che deve rispettare un sistema di cifratura, con il fine di mostrare che essi sono rispettati da S^2C.

1. Il sistema dovrebbe essere, se non teoricamente, impossibile da rompere almeno in pratica.

2. La pubblicazione dei dettagli del sistema non dovrebbe compromettere la sicurezza della comunicazione.

3. La chiave dovrebbe essere facilmente ricordabile (senza che sia necessario annotarla) e facilmente sostituibile.

4. Il messaggio cifrato dovrebbe poter essere trasmesso con un telegrafo.

5. L'apparato per le operazioni di cifratura dovrebbe essere trasportabile ed utilizzabile da una singola persona.

6. Il sistema dovrebbe essere semplice e non richiedere la conoscenza di una lunga lista di regole né sforzo mentale.

Per ognuno dei vincoli di cui sopra accenniamo, brevemente, al perché S^2C lo rispetti.

1. La procedura S^2C effettua una cifratura che, nel caso in cui la key image e la user key siano utilizzate una sola volta, è perfettamente sicura (one-time pad). Nel caso si usi più volte la stessa user key, attenendosi al protocollo descritto nel paragrafo V.5, la cifratura è impossibile da rompere in pratica.

2. La pubblicazione dei dettagli implementativi dell'algoritmo non inficia minimamente la sicurezza del sistema, tant'è vero che questa tesi si dilunga su tali dettagli. L'unica cosa che deve restare segreta è una chiave condivisa da sender e receiver, cioè la user key.

3. La chiave utente, come dice il nome stesso, è scelta dall'utente come egli meglio crede, quindi molto probabilmente questi sceglierà una parola o una frase facilmente ricordabile. La key image e la user key possono essere facilmente sostituite ad ogni nuovo messaggio.

4. Siccome il telegrafo, senza far torto a Kerckhoffs, non lo usa più nessuno, questo vincolo risulta soddisfatto perché la procedura S^2C può essere adoperata per inviare messaggi tramite qualsiasi canale digitale, in particolare Internet (che si può vedere come un "moderno telegrafo").

5. L'algoritmo S^2C è implementato tramite software, che può girare su qualsiasi computer ed essere usato da un utente singolo.

6. Il sistema S^2C è stato implementato prevedendo una interfaccia grafica (GUI) che permette a chiunque di utilizzarlo senza alcuno "sforzo mentale". L'utente non deve fare altro che scegliere messaggio segreto, key image e cover image; il resto lo fa S^2C.

Concludiamo, forti di quanto è stato scritto, che S^2C rispetta i requisiti di Kerckhoffs, quindi è un sistema crittografico valido a tutti gli effetti.

Nota conclusiva *Nella prefazione del presente capitolo si è detto che S^2C realizza le due operazioni di steganografia e crittografia allo stesso tempo. La dimostrazione di questa affermazione è molto banale e potrebbe essere sottintesa, in ogni caso, per completezza essa si riporta a seguire.*
L'algoritmo S^2C presentato nel paragrafo IV.9.3 è usato per cifrare il messaggio segreto, quindi esegue una operazione crittografica. Questo stesso algoritmo funziona sostituendo i coefficienti DCT di una immagine JPEG, quindi esegue una operazione steganografica. Si conclude che S^2C, per mezzo dello stesso algoritmo e quindi contemporaneamente, permette di fare crittografia e steganografia.

Capitolo VI

Conclusioni

Capitolo VI

Conclusioni

Le conclusioni finali sono racchiuse in questo breve capitolo, il cui fine è quello di tirare le somme del lavoro fin qui presentato, evidenziando ciò che rende unico S^2C rispetto agli algoritmi che già sono conosciuti in letteratura. In più vengono presentate due possibili applicazioni di S^2C, per poi prospettare gli sviluppi futuri di questa tesi.

VI.1 Considerazioni finali

Le pagine precedenti hanno mostrato come sia possibile costruire un sistema che sia, al tempo stesso, steganografico e crittografico. Ciò è possibile "prendendo in prestito" gli elementi caratteristici della stereo visione: la coppia di immagini stereo e la mappa di disparità. L'innovazione principale di questa tesi consiste nell'uso inedito che si fa di tali elementi, la stereo visione andando ad assumere il ruolo di trait d'union tra steganografia e crittografia. Da questa unione nasce S^2C, una procedura che mappa una delle due immagini stereo sulla stego image, la mappa di disparità sul messaggio segreto e la restante immagine stereo sulla key image (parte bassa della figura VI.1).
Il termine key image indica un elemento estraneo alla steganografia, ma ben inquadrabile all'interno della teoria crittografica, considerando la key image come la chiave del noto algoritmo one-time pad (in alto a destra nella figura VI.1).

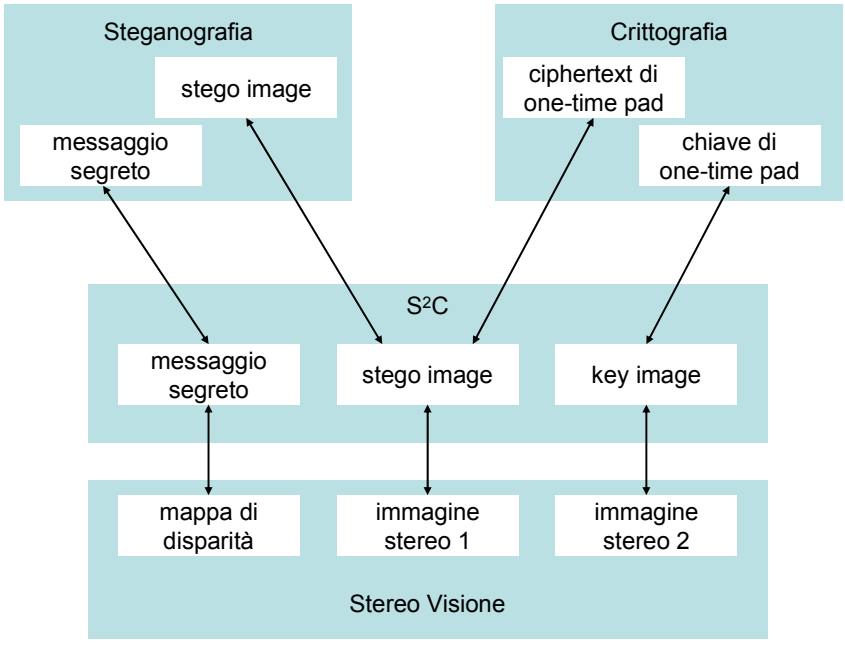


Figura VI.1 – S^2C in rapporto a steganografia, crittografia e stereo visone

S^2C permette di fare contemporaneamente steganografia e crittografia proprio grazie all'intervento unificatore della stereo visione, che dà facoltà di unire stego image (propria della steganografia) e key image (propria della crittografia) considerandole una coppia stereo.
Come è stato approfonditamente discusso nel capitolo IV, che rappresenta il nucleo di questa tesi, la procedura S^2C viene identificata nell'inversione dell'algoritmo di correlazione (capitolo II). Si tratta, va detto, di una inversione sui generis a causa di alcune limitazioni imposte dalla steganografia e trattate sempre nel capitolo IV. Quest'ultimo contiene l'enunciazione delle caratteristiche che rendono unico S^2C:

4. l'algoritmo S^2C permette di effettuare steganografia senza alterare alcuna delle principali caratteristiche statistiche di una immagine (descritte nel precedente paragrafo IV.9);

5. S^2C consente di cifrare il messaggio *m* con un algoritmo equivalente al cifrario di Vernam [22], da cui deriva one-time pad;

6. S^2C compie le azioni descritte ai precedenti punti **allo stesso tempo**.

Le dimostrazioni di quanto affermato riempiono il capitolo V insieme alla verifica che S^2C, in quanto algoritmo crittografico, rispetti effettivamente i desiderata di Kerckhoffs. Sempre nel capitolo V, si trova uno scenario d'uso di S^2C che permette di capire come esso sia in grado di superare i problemi di one-time pad nella diffusione e nella gestione della chiave. Proprio il confronto con one-time pad ci permette di asserire quanto segue:

Definizione VI.1 *S^2C è un algoritmo steganografico in grado di resistere ad attacchi visuali e statistici ed è, al tempo stesso, un algoritmo crittografico che permette di cifrare il messaggio mediante una tecnica con garanzia di segretezza perfetta.*

In altre parole, S^2C si pone a cavallo tra la crittografia e la steganografia, offrendo in più della steganografia la possibilità di cifrare il messaggio senza pre-elaborazione, ed eliminando, rispetto alla crittografia, la riconoscibilità del messaggio inviato.

VI.2 Possibili applicazioni

Tra le possibili applicazioni di S2C, ne verranno presentate due:

- un client di posta per l'invio di e-mail sicure,

- un sistema, basato sul paradigma S^2C, per la codifica e decodifica steganografica in formati video quali M-JPEG ed MPEG.

Il client di posta è già realtà, essendo stato sviluppato con il nome di *S2C Mail*, in contemporanea alla stesura di queste pagine. L'estensione di S^2C, dalle immagini JPEG ai formati video M-JPEG ed MPEG, deve ancora essere effettivamente implementata, anche se si può dire, fin d'ora, che tale implementazione non richiede nessun particolare sforzo.

VI.2.1 S2C Mail

S2C Mail è un client di posta elettronica basato sul protocollo riportato in figura V.10. Il programma è scritto in Java, è composto da oltre 30 classi e conta più di 9000 linee di codice. È stato lungamente testato ed è perfettamente impiegabile per scopi non solo didattici. A seguire, uno scenario d'uso di S2C Mail.

Esempio VI.1 *Alice e Bob posseggono entrambi una copia di S2C Mail e condividono una chiave, che può essere una qualsiasi password p.*
Inizializzazione: *Alice invia a Bob una immagine I, per mezzo di un client di posta tradizionale oppure la colloca in un sito che anche Bob conosce (e da cui questi può scaricarla). I non sarà decodificata, quindi può essere una immagine senza nessun contenuto occulto. Una volta che sia Bob sia Alice sono in possesso di I, il protocollo S^2C può partire.*
Lato sender: *Alice apre S2C Mail e digita un messaggio di testo (oppure sceglie un file) da inviare, sia m tale messaggio. Il programma chiede ad Alice di scegliere una immagine C_1, che sarà la cover image; Alice sceglie una qualsiasi immagine tra quelle di cui dispone. A questo punto, il software richiede di fornire la key image ed una password: Alice deve scegliere I e digitare la password p.*
Compiuti questi passi, S2C Mail produce una immagine S_1 che Alice può inviare, come allegato di posta, a Bob.
Lato receiver: *Bob attiva S2C Mail, scarica S_1 dal suo server di posta e chiede al programma di leggere il messaggio segreto. S2C Mail invita Bob a scegliere una chiave (che sarà I) ed una password (p). Fatto questo, il programma mostra a Bob il messaggio m.*
Se, ora, Bob vuole rispondere ad Alice, salta la fase di inizializzazione ed usa come immagine I il file S_1 che ha appena ricevuto; quindi segue il protocollo lato sender. □

VI.2.2 Codifica e decodifica video

L'algoritmo S^2C è stato progettato per immagini JPEG. Poiché esso utilizza come cover e key due immagini qualsiasi, può essere adoperato per la modifica di file video che adottino una compressione "simile" a JPEG. È questo il caso dei due formati video M-JPEG (che sta per Motion-JPEG) ed MPEG (Moving Picture Experts Group). Il primo, M-JPEG, consiste, semplificando, in una sequenza di immagini compresse con JPEG e trasmesse con un rate adeguato. Lo standard MPEG, giusto per dare una idea, si basa sullo stesso concetto, ma permette di ottenere una compressione maggiore poiché non è una semplice sequenza di immagini JPEG

(chiamate in questo standard frames di tipo I), essendo queste ultime intervallate da immagini ricostruibili con minore informazione (frames P e B) [41].
Le figure VI.2 e VI.3 mostrano la fase di codifica e la fase di decodifica per il formato M-JPEG. Discorso analogo vale per MPEG, considerando solo i frames di tipo I [41].

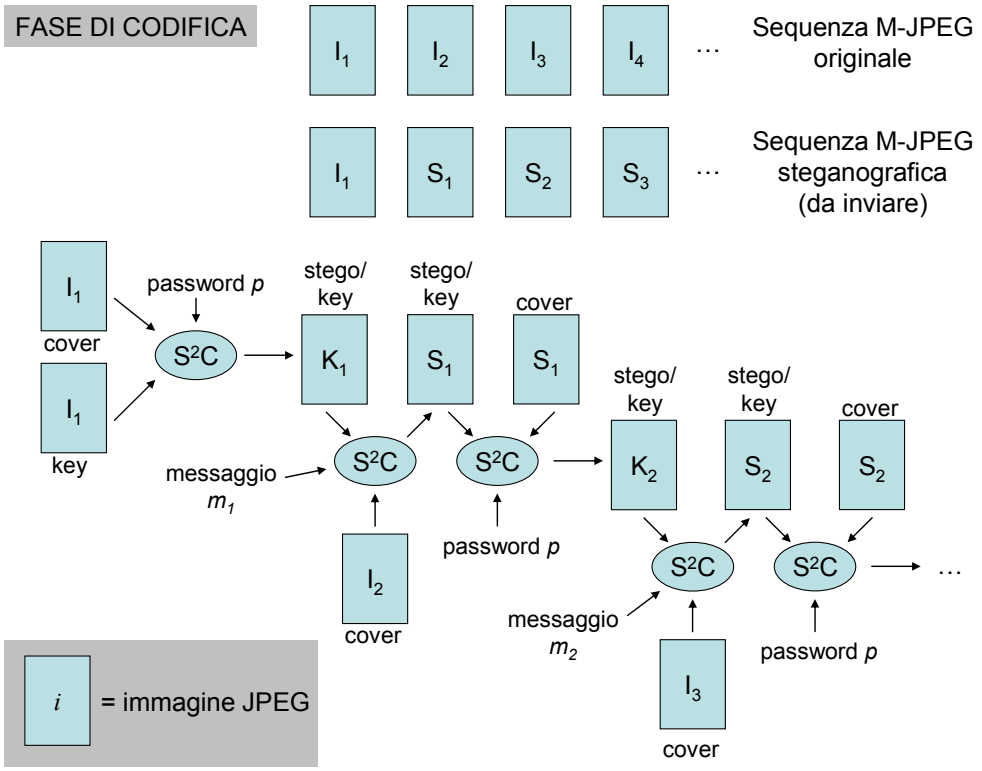


Figura VI.2 – S^2C per video, fase di codifica

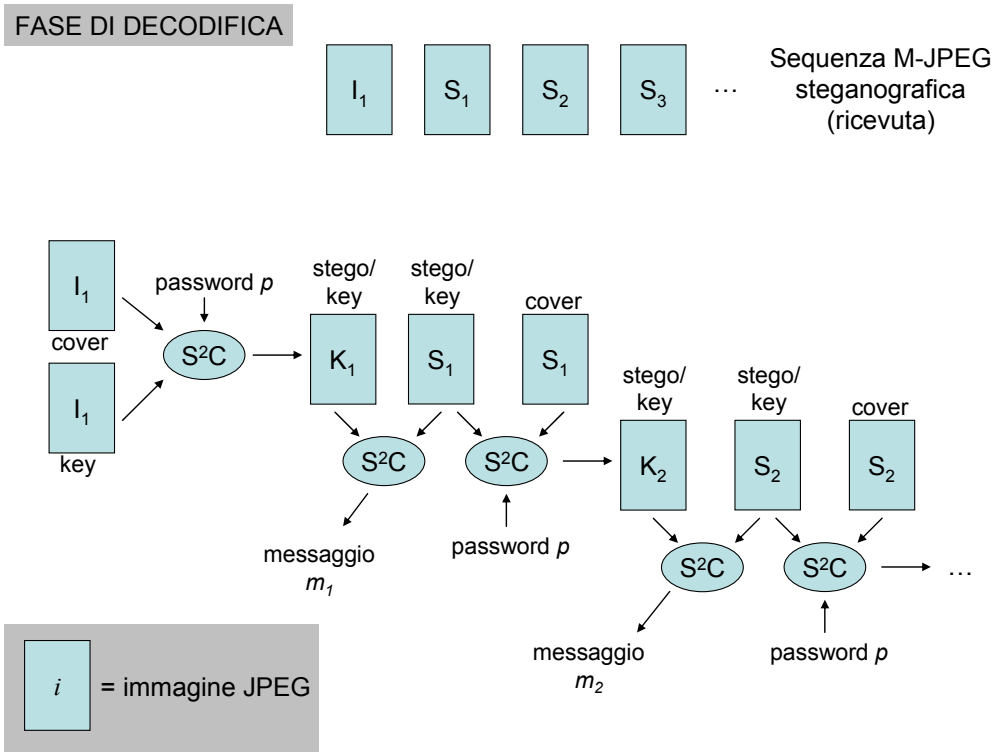


Figura VI.3 – S^2C per video, fase di decodifica

Nelle figure VI.2 e VI.3 si può notare che le stesse immagini assumono prima il ruolo di stego image e quindi quello di key image (dizione stego/key nelle figure VI.2 e VI.3). La possibilità di assegnare ad ogni immagine questo duplice ruolo, consente al sistema S^2C di non obbligare sender e receiver all'invio di due immagini (la stego e la key) per ogni messaggio. Il paradigma risultante è un vero e proprio one-time pad video. Inoltre, l'uso di un file video, oltre a permettere l'inserimento di un volume di informazioni steganografiche enormemente superiore rispetto ad una singola immagine, consente di evitare la fase di inizializzazione illustrata nell'esempio VI.1.

VI.3 Sviluppi futuri

Uno sviluppo futuro per il lavoro svolto in questa tesi riguarda la ricerca di un attacco che possa battere S^2C. Così come la fase di validazione di un software non può essere effettuata da chi lo ha scritto, chi ha sviluppato S^2C non potrebbe ricercare correttamente un modo di batterlo; ciò vuol dire che questo possibile sviluppo deve essere portato avanti da una persona diversa da chi ha scritto queste pagine.
Potrebbe essere interessante provare ad attaccare S^2C con una tecnica simile a quella proposta da Fridrich [38] per battere F5. Per motivi di tempo, non è stato possibile effettuare questa verifica. Va notato, comunque, che la fase di embedding di F5, così come la fase di decodifica, è molto diversa da S^2C, quindi un attacco che funzioni per F5 potrebbe non funzionare su S^2C. In particolare, l'attacco di Fridrich si basa sulla modifica statistica dei coefficienti 0 ed 1 perpetuata da F5; S^2C non modifica minimamente il numero totale (la statistica) di questi coefficienti.
Un secondo sviluppo potrebbe riguardare l'estensione dell'algoritmo S^2C per la modifica steganografica di immagini in formato JPEG 2000, lavorando, in un qualche modo, sulla trasformata Wavelet piuttosto che sulla DCT.

Bibliografia

Bibliografia

[1] Mehdi Kharrazi, Husrev T. Sencar and Nasir Memon, *Image Steganography: Concepts and Practice*, WSPC/Lecture Notes Series, April 2004.

[2] Sabu M. Thampi and Dr. K. Chandra Sekaran, *Steganography Based WWW Distributed Image Retrieval with Mobile Agents*.

[3] Ross J. Anderson and Fabien A.P. Petitcolas, *On The Limits of Steganography*, IEEE Journal of Selected Areas in Communications, 16(4):474-481, May 1998.

[4] Niels Provos and Peter Honeyman, *Hide and Seek: An Introduction to Steganography*, IEEE SECURITY & PRIVACY, 2003.

[5] Max Weiss, *Principles of Steganography*.

[6] Neil F. Johnson and Sushil Jajodia, *Exploring Steganography: Seeing the Unseen*, February 1998.

[7] Yeuan-Kuen Lee and Ling-Hwei Chen, *Secure Error-Free Steganography For JPEG Images*, International Journal of Pattern Recognition and Artificial Intelligence, Vol. 17, No. 6 (2003) 967-981.

[8] Maurizio Migliore, *Steganografia*, slides, 2002

[9] Jonathan Cummins, Patrick Diskin, Samuel Lau and Robert Parlett, *Steganography And Digital Watermarking*, 2004.

[10] Daniel L. Currie, III and Cynthia E. Irvine, *Surmounting the Effects of Lossy Compression on Steganography*.

[11] Andreas Westfeld and Gritta Wolf, *Steganography in a Video Conferencing System*, David Aucsmith (Ed.): Information Hiding 1998, LNCS 1525, pp. 32-47, 1998.

[12] James Caldwell, *Steganography*, Software Engineering Technology, 2003.

[13] Nicholas J. Hopper John Langford Luis von Ahn, *Provably Secure Steganography*, CMU-CS-02-149, September 2002.

[14] Weiming Zhang and Shiqu Li, *On the Unicity Distance of Stego Key*, Journal Of Latex Class Files, Vol. 1, no. 11, November 2002.

[15] Gary C. Kessler, *Steganography: Hiding Data Within Data*, September 2001.

[16] G. J. Simmons, *The prisoners' problem and the subliminal channel*, in Advances in Cryptology: Proceedings of Crypto 83 (D. Chaum, ed.), pp. 51-67, Plenum Press, 1984.

[17] Domenico Bloisi, *Tecniche di correlazione per la stereo visione*, tesi, 2004.

[18] A. Marchetti Spaccamela, *Lucidi del corso di Crittografia e sicurezza delle reti*, slides, 2004.

[19] A.Westfeld and A. Pfitzmann, *Attacks on Steganographic Systems*, Proc. Information Hiding 3rd Int'l Workshop, Springer Verlag, 1999, pp. 61-76.

[20] A. Kerckhoffs, *La Cryptographie Militaire*, Journal des Sciences Militaries, 9th series, IX (Gennaio 1883) pp. 5-38; (Febbraio 1883) pp. 161-191.

[21] *Steganografia*, http://www.tonycrypt.com/Crittografia/Stegano.htm

[22] A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[23] Riccardo Stagliano', *Steganografia, la scienza più amata dai terroristi*, la Repubblica.it, Tecnologie e Internet, 30 ottobre 2001.

[24] SNOW, Steganographic nature of whitespace, http://www.darkside.com.au/snow

[25] Giampietro Allasia e Umberto Cerruti, *Crittografia Invisibile*, pubblicato su Tuttoscienze, 21 aprile 1999.

[26] Claudio Agosti, *Steganografia, l'arte della scrittura nascosta*, slides, 2003.

[27] J. Fridrich, M. Goljan, R. Du, *Steganalysis Based on JPEG Compatibility*.

[28] N. F. Johnson, Z. Duric and S. Jajodia, *Information Hiding: Steganography and Watermarking – Attacks and Countermeasures*, Kluwer Academic Publishers, Boston Dodrecht London (2000).

[29] N. Provos, P. Honeyman, *Detecting Steganographic Content on the Internet*, CITI Technical Report 01-11, 2001.

[30] Jack Kelley, *Terror groups hide behind Web encryption*, USA Today, febbraio 2001.

[31] Declan McCullagh, *Secret Messages Come in .Wavs*, Wired News, febbraio 2001.

[32] Serena Longhini, *Procedure steganografiche e protocolli speciali per l'occultamento e la trasmissione dei dati*, tesi, 2003.

[33] C. E. Shannon, *Communication theory of secrecy system*, Bell Syst. Tech. J., vol. 28, pp. 656-715, 1949.

[34] Paolo Cavallo, *Tecniche di steganografia per immagini codificate tramite la dwt ed analisi comparativa con steganografia per immagini jpeg*, tesi, 2001.

[35] A. Westfeld, *F5—A Steganographic Algorithm: High Capacity Despite Better Steganalysis*, Proc. 4th Int'l Workshop Information Hiding, Springer-Verlag, 2001, pp. 289-302.

[36] C. Cachin, *An Information-Theoretic Model for Steganography*, Cryptology ePrint Archive, Report 2000/028, 2002.

[37] G. K. Wallace, *The JPEG Still Picture Compression Standard*, IEEE Trans. Consumer Electronics, Vol. 38, No 1, Feb. 1992.

[38] J. Fridrich, M. Goljan and D. Hogea, *Steganalysis of JPEG images: Breaking the F5 algorithm*, in 5th International Workshop on Information Hiding, 2002.

[39] La Biblioteca di Repubblica Grandi Opere di Cultura UTET, *L'Enciclopedia*, volume 5, 2003.

[40] N. Provos, *Defending Against Statistical Steganalysis*, Proc. 10th Usenix Security Symp., Usenix Assoc., 2001, pp. 323-335.

[41] *MPEG Digital Video-Coding Standards*, IEEE Signal Processing Magazine, September 1997, pp.82-100.

Finito il 20 maggio 2006.

Alla prossima.