



UNIVERSITA' DEGLI STUDI DELLA BASILICATA

DIPARTIMENTO DI MATEMATICA, INFORMATICA ED ECONOMIA

Corso di Laurea in Scienze e Tecnologie Informatiche

Steganografia su PDF

Candidato:
Rocchina Romano
Matricola 41003

Relatore:
Ing. Domenico Daniele Bloisi

"Il posto migliore per nascondere qualsiasi cosa è in piena vista."
(Edgar Allan Poe, La lettera rubata)

Abstract

La seguente Tesi ha l'obiettivo di fornire una nuova tecnica steganografica per l'inserimento e l'estrazione di informazioni in/da documenti testuali, in particolare utilizzando il formato PDF ("Portable Document Format"). La Steganografia è la materia che si occupa di nascondere messaggi segreti all'interno di file contenitori (come ad esempio, immagini, video, tracce audio, testo, ecc...), in modo tale che eventuali osservatori non siano in grado di individuarli. Per prima cosa, è necessario individuare il file contenitore, detto "cover PDF", cioè il file nel quale inserire il messaggio segreto. Poi sarà necessario utilizzare opportune tecniche non invasive per inserire il messaggio nel cover medium. Alla fine di questo processo steganografico, si otterrà il cosiddetto "stego PDF", il quale dovrà essere il più possibile simile al "cover PDF" e dovrà contenere l'informazione steganografica, che potrà essere estratta dal destinatario.

This Thesis aims to provide a novel steganographic technique for embedding and extracting information inside/from text documents, in particular using the PDF format ("Portable Document Format"). Steganography is the art of communicating in a way which hides the very existence of the communication (Markus Kuhn - 1995-07-03). First of all, it is necessary to choose the cover file, called "cover PDF", that is the file in which to embed the secret message. At the end of this steganographic process, the "stego PDF" will be obtained, which must be as similar as possible to the "cover PDF" and must contain the steganographic information, can be extracted by the receiver.

RINGRAZIAMENTI

Per prima cosa vorrei ringraziare chi mi ha dato la possibilità di poter realizzare questo progetto: l' *Ing. Domenico Daniele Bloisi*.
Grazie Prof. per l'opportunità e la fiducia dimostratami.

In secondo luogo, vorrei ringraziare la mia famiglia, mia *sorella Antonella*, *mio padre Antonio* e *mia madre Maria*, i quali hanno avuto la pazienza di sopportarmi e di appoggiarmi in questi mesi. Grazie per avermi dato la possibilità di intraprendere questo percorso universitario.

Un ringraziamento speciale va al mio *amico Canio*: grazie per tutte le avventure e disavventure passate insieme. Grazie per essermi stato sempre accanto e per esserti dimostrato l'amico "sincero", quale sei.

Infine, vorrei ringraziare me stessa: grazie per non esserti mai arresa, nonostante le difficoltà e nonostante alcuni ti abbiano fatto capire, anche indirettamente, che non ce l'avresti mai fatta. Sappiate che mi avete dato un motivo in più per andare avanti (non che non ce l'avessi già!!!).

Inoltre, per non dimenticare nessuno, vorrei lasciare un *grazie "generale"* per chi mi è stato accanto in questo percorso e mi ha incoraggiata affinché io non mollassi.

Perché ricordatevi sempre questa cosa: io "**barollo, ma non mollo mai**"!!!
Grazie di cuore a tutti!!

Rocchina.

INDICE

1 Introduzione	3
1.1 Il Problema	3
1.2 La Crittografia	4
1.2.1 Terminologia utilizzata	4
1.2.2 Crittografia a chiave simmetrica e crittografia a chiave pubblica	4
1.3 La Steganografia	5
1.3.1 La steganografia nella storia	5
1.3.2 La steganografia nell'arte	9
1.3.3 Terminologia utilizzata	13
1.3.4 Il problema del prigioniero	14
1.3.5 Proprietà di un sistema steganografico	14
1.3.6 Classificazione delle tecniche steganografiche	15
1.4 Attacco alla Steganografia: la Steganalisi	17
1.4.1 Principio di Kerckhoffs	17
1.5 Steganografia vs Crittografia	18
1.6 Le motivazioni	20
1.7 I contributi	21
1.8 Struttura della Tesi	21
2 Stato dell'Arte delle Tecniche Steganografiche	23
2.1 La Steganografia sulle Immagini	23
2.1.1 Tecnica di inserimento del bit meno significativo(LSB)	23
2.1.2 Mascheramento e filtraggio	25
2.1.3 Algoritmi e trasformazioni	25
2.1.4 Tecniche di steganalisi sulle immagini	27
2.1.5 Alcune applicazioni della steganografia sulle immagini	29
2.2 La Steganografia sul Testo	32
2.2.1 Steganografia linguistica	33
2.2.2 Generazione casuale e statistica	34
2.2.3 Steganografia basata sul formato	34
2.2.4 Applicazioni attualmente esistenti di steganografia sul testo	35
3 Metodi e Implementazione	39
3.1 Il Formato PDF ("Portable Document Format")	39
3.1.1 Gli oggetti del formato PDF	40
3.1.2 L'oggetto NULL	41
3.1.3 L'oggetto indiretto e l'oggetto diretto	41
3.1.4 Riferimenti agli oggetti	42
3.1.5 Struttura di un file PDF	42

3.1.6	Struttura di un documento PDF	43
3.1.7	Resources dictionaries (o dizionari delle risorse)	44
3.1.8	XObject	45
3.2	Steganografia su PDF	46
3.2.1	Processo di embedding	46
3.2.2	Processo di estrazione	49
3.3	Aspetti Implementativi	50
3.3.1	Processo di embedding	51
3.3.2	Processo di estrazione	59
4	Risultati Sperimentali	63
4.1	Dataset per gli Esperimenti	63
4.2	Caratteristiche dello Stego PDF	64
4.3	Contenuto Copiato in un Documento MS-Word	66
4.4	Unione di più Stego PDF	67
4.5	Ritaglio di un File PDF	68
5	Conclusioni	71
5.1	Risultati Raggiunti	73
5.2	Codice Sorgente ed Utilizzo dell'Applicazione	73
5.3	Sviluppi Futuri	74
Bibliografia		76

Capitolo

1

INTRODUZIONE

1.1 Il Problema

La Steganografia è la scienza che si occupa, non tanto di nascondere il contenuto di una comunicazione segreta, quanto quello di nascondere l'esistenza stessa di una comunicazione segreta. A differenza della Crittografia, è una scienza poco conosciuta, ma che allo stesso tempo può essere utilizzata per garantire la riservatezza, la resistenza alla manomissione e, soprattutto, la sicurezza delle informazioni scambiate tra due o più persone. Affinché ci possa essere uno scambio di informazioni steganografiche, è necessario scegliere con cura l'oggetto *cover*, ovvero il mezzo di trasporto, che alla fine del processo steganografico, conterrà l'informazione segreta da inviare successivamente al destinatario. L'oggetto *cover* può essere un qualunque file multimediale, come ad esempio un file testuale, un'immagine digitale, una traccia audio oppure un videoclip. Alla fine di questo processo, detto *processo di embedding*, si ottiene il cosiddetto oggetto *stego*, che contiene l'informazione steganografica e che deve essere il più possibile simile all'oggetto *cover*, in modo tale che eventuali osservatori non siano in grado di rilevare possibili modifiche; infatti, se così non fosse, la comunicazione steganografica può considerarsi fallita.

Bisogna prestare molta attenzione al fatto che in alcuni Paesi la Steganografia è considerata una pratica illegale. Tuttavia, l'obiettivo principale di questa Tesi è quello di utilizzare la Steganografia al fine di garantire la privacy di informazioni strettamente personali, come ad esempio nel caso dei referti medici, per i quali si ha la necessità di mantenere segreta l'identità del paziente, oppure per questioni strettamente legate alla protezione del copyright. Verrà illustrata una nuova tecnica steganografica, che permetterà di inserire e di estrarre informazioni segrete, utilizzando il formato PDF ("Portable Document Format"). Prima di procedere nei dettagli di questa nuova tecnica steganografica, forniremo una panoramica generale sulla Crittografia e sulla Steganografia. In particolare, ci soffermeremo sulle tecniche steganografiche attualmente esistenti di Steganografia sulle immagine e sul testo.

1.2 La Crittografia

¹In questo paragrafo verranno fornite solamente alcune nozioni di base ed una panoramica generale sulla Crittografia, dal momento che l'argomento principale della seguente tesi è la Steganografia, sulla quale verrà fatta una trattazione più dettagliata.

Il termine Crittografia deriva dal greco "kryptòs" e "gráphein", che significano rispettivamente "nascosto" e "scrivere". Quindi, la crittografia è l'arte e la scienza che si occupa di mantenere segreta una conversazione tra due o più persone, in modo tale che eventuali "nemici" non siano in grado di decifrare le informazioni scambiate. In pratica, il messaggio è ben visibile, ma appare insignificante ed incomprensibile agli occhi di chi non possiede la "chiave" giusta per poter decifrare il messaggio.

1.2.1 Terminologia utilizzata

Testo in chiaro(plaintext o cleartext): il testo in chiaro è un messaggio che si presenta nella sua forma naturale, vale a dire che può essere letto e compreso da tutti. Quindi, è un messaggio che non ha subito alcun processo crittografico.

Testo cifrato(ciphertext): il testo cifrato è un messaggio che, al contrario, ha subito un processo crittografico. Esso si ottiene a partire dal testo in chiaro, dopo aver applicato una chiave di crittografia.

Cifratura(Encryption): è il processo che, a partire dal testo in chiaro, consente di ottenere il testo cifrato.

Decifratura(Decryption): è il processo che, a partire dal testo cifrato, consente di recuperare il testo in chiaro.

Crittoanalisi: è, in pratica, una "attacco" alla crittografia. Infatti, tenta di rilevare eventuali punti deboli nel sistema crittografico utilizzato e di risalire al messaggio in chiaro. Infatti, la comunicazione crittografica fallisce proprio nel momento in cui si riesce a venire a conoscenza della chiave segreta utilizzata.

Chiave(o key): la chiave è un codice alfanumerico che, se applicato al testo in chiaro, ci fornisce il testo cifrato e, se applicato correttamente al testo cifrato, ci permette di ottenere il testo in chiaro.

1.2.2 Crittografia a chiave simmetrica e crittografia a chiave pubblica

A seconda della tipologia di chiave utilizzata, possiamo fare una distinzione tra Crittografia a chiave simmetrica e Crittografia a chiave pubblica.

La **Crittografia a chiave privata o simmetrica** utilizza una sola chiave sia per la cifratura, sia per la decifratura. In pratica, sia k la chiave simmetrica utilizzata. Supponiamo che Alice voglia inviare un messaggio a Bob, utilizzando la chiave k. Per prima cosa, Alice dovrà applicare la chiave k al testo in chiaro, ottenendo così il testo cifrato, che invierà successivamente a Bob. Se Bob vorrà recuperare il messaggio inviatogli da Alice, dovrà semplicemente applicare la stessa chiave (al testo cifrato) che Alice ha utilizzato per cifrare il messaggio in chiaro. Quindi, Alice e Bob utilizzano la stessa chiave. Proprio per questo motivo, la Crittografia a chiave privata è anche chiamata Crittografia a chiave simmetrica. La Crittografia a chiave simmetrica, però, presenta il seguente svantaggio: Alice e Bob per poter comunicare devono prima concordare in segreto una chiave da poter utilizzare per la cifratura e per la decifratura. Inoltre, qualunque "ficcanaso" può venire a conoscenza della chiave e risalire al messaggio in chiaro.

Non sempre, però, le due parti possono concordarsi a priori sulla chiave da utilizzare e,

¹il seguente paragrafo fa riferimento a [4], [19], [21], [13].

pertanto, il problema che si presenta è il seguente: è possibile cifrare i messaggi senza aver concordato prima una chiave? La soluzione al seguente problema ci venne data da Diffie e Hellman nel 1976 e si è dimostrata particolarmente utile nello sviluppo della **Crittografia a chiave pubblica**, utilizzata soprattutto per l'autenticazione (ovvero provare l'identità di una persona) e la firma digitale.

L'idea alla base della Crittografia a chiave pubblica è la seguente: supponiamo che Alice voglia comunicare con Bob. A differenza della Crittografia a chiave simmetrica, in questo caso Bob non possiede una chiave simmetrica, bensì due: una pubblica k_+ , che è nota a tutti (anche ad eventuali "nemici" o "ficcanasi") ed una privata k_- , che conosce solo lui. Alice, prima di mandare il messaggio, applica al testo in chiaro la chiave pubblica k_+ di Bob. Bob, per poter ottenere il messaggio in chiaro, deve prima applicare la sua chiave pubblica k_+ al messaggio cifrato e successivamente applicare la sua chiave privata k_- al messaggio precedentemente decifrato con la sua chiave pubblica k_+ . Quindi, in pratica, se il messaggio in chiaro è m , avremo

$$m = k_-(k_+(m))$$

Solo a questo punto Bob potrà realmente recuperare il messaggio che gli è stato inviato da Alice. In ogni caso, il risultato non cambia se Bob decidesse di utilizzare prima la sua chiave privata e poi la sua chiave pubblica, quindi

$$m = k_+(k_-(m))$$

Tuttavia, è fondamentale che la chiave privata rimanga ignota, in modo tale da impedire ad un eventuale "nemico" o "ficcanaso" di scoprire il messaggio.

1.3 La Steganografia

²Il termine Steganografia deriva dal greco "stèganos" e "gráphein", che significano rispettivamente "coperta" e "scrittura"; pertanto, letteralmente significa "scrittura coperta". Quindi, la steganografia è l'arte e la scienza che si occupa non tanto di nascondere il contenuto di una comunicazione segreta, quanto l'esistenza stessa della comunicazione agli occhi di eventuali "nemici". Quindi, il contenuto della comunicazione non è in piena vista, ma è come se fosse invisibile. Infatti, solamente il destinatario è a conoscenza di ciò.

1.3.1 La steganografia nella storia

Le storie di Erodoto

La Steganografia non si è sviluppata di recente, ma affonda le sue radici già nel 400 a.C. nell'antica Grecia. Erodoto nei suoi racconti parla di due tecniche, che possono essere considerate delle tecniche steganografiche. La prima consisteva in tavolette di legno ricoperte di cera, sulle quali venivano incisi dei messaggi segreti. Infatti, Erodoto, a tale proposito, racconta di Demerstus, un esiliato greco in una città persiana, il quale era venuto a conoscenza del fatto che Xerxes, re dei Persiani, voleva invadere la Grecia. Egli, nonostante si trovasse in esilio, trovò uno stratagemma per avvisare i suoi compatrioti: prese una tavoletta di legno, vi incise il messaggio, dopodiché ricoprì il tutto con della cera. Questa tavoletta, così come si presentava, appariva innocua agli occhi dei Persiani e fu per questo motivo che riuscì ad oltrepassare le ispezioni ed arrivare in Grecia. In un'altra storia, invece, racconta di Histaeus, un sovrano persiano, che fece rasare a zero i capelli di un suo schiavo fidato, in modo da poter tatuare un messaggio sul suo cranio. Una volta che i capelli furono ricresciuti, inviò lo schiavo verso la sua destinazione con la sola istruzione di tagliarsi nuovamente i capelli e mostrare, quindi, il messaggio nascosto.

²il seguente paragrafo fa riferimento a [19], [4], [16]

L'antica Cina

Nell'antica Cina, invece, si dipingeva il messaggio nascosto su striscioline di seta finissima, che venivano successivamente appallottolate e coperte di cera. Per evitare che i messaggi fossero intercettati, le palline erano inghiottite dal messaggero.

Inchiostri invisibili

Gli antichi Romani utilizzavano sostanze come succhi di frutta, latte ed urina per scrivere messaggi segreti. Quando queste sostanze venivano riscaldate, i messaggi diventavano leggibili e, di conseguenza, potevano essere letti.

Le griglie di Cardano

Un altro strumento, usato a scopi steganografici, erano le famose Griglie di Cardano. Esse consistevano in fogli di materiale rigido, nei quali venivano ritagliati dei fori rettangolari ad intervalli irregolari. Questa griglia veniva appoggiata su un foglio di carta bianca, il messaggio segreto veniva scritto nei buchi, dove ciascun buco poteva contenere una o più lettere, dopodiché si toglieva la griglia e si cercava di completare la scrittura del resto del foglio, in modo tale da ottenere un messaggio di senso compiuto. Una volta fatto ciò, il messaggio veniva inviato al destinatario, il quale, per poter risalire al testo nascosto, doveva applicare sul messaggio ricevuto una copia esatta della griglia originaria.

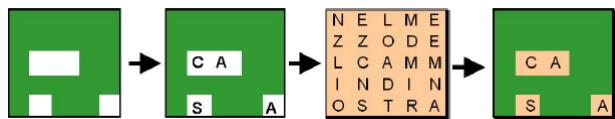


Figura 1.1: Esempio di Griglia di Cardano preso da [19].

Tritemio

Il termine Steganografia compare per la prima volta nel 1499 grazie ad un'opera di Tritemio, appunto, intitolata "Steganographia". Si tratta di una trilogia che per anni circolò privatamente in forma di manoscritto; venne pubblicata solamente nel 1606, ma la sua diffusione durò bene poco, dal momento che nel 1609 esse fu collocata nella lista ufficiale dei libri proibiti. I primi due libri contengono numerosi metodi per nascondere messaggi all'interno di altre scritture più composite e, talvolta, prive di significato. Ad esempio, nella frase

"Paramiesel oshurmi delmuson thafloin peano charustrea melany lyamunto..."

la prima parola specifica il sistema crittografico che deve essere usato, dopodiché a partire dal secondo termine, leggendo una lettera no ed una sì e saltando una parola, si ottiene la seguente espressione:

"Sum tali cautela ut..."

La terza parte della trilogia è apparentemente un libro di astronomia occulta: è composta da tabelle di numeri, le cui colonne sono sovrastate da simboli zodiacali e planetari, che appaiono come dati astronomici.

G.	R.	G.	R.	G.	R.
Her.1.	Her.2.	her.3.	grad.	punct.	her.1.
640	635	22	25	634	632
642	646	647	3	646	32
644	25	646	2	648	640
646	640	632	4	632	630
648	645	634	4	639	644
650	642	30	1	647	639
			5		dat.
G.	R.	G.	R.	G.	R.
hor.2.	hor.3.	Y			
632	632	650			
640	640	640			
642	633	646			
647	632	639			
638	632	650			
639	640	626			
		R.			
		9			
		4			

Figura 1.2: Terza parte della trilogia "Steganographia" presa da [19].

Fu Jim Reeds, un matematico crittoanalista, che riuscì a svelare lo schema usato da Tritemio, convinto che il terzo libro nascondesse dei dati in codice. Infatti, la sua ricerca portò alla conclusione che gli schemi di Tritemio altro non erano che cifrari di sostituzione numerica con equivalenti numerici multipli per ogni lettera del testo in chiaro.

Nella seguente figura forniamo una tabella originale alla base della cifratura di Tritemio:

T	H	S	C	B	A							
01	02	03	04	05	06							
07	08	09	10	11	12							
26	27	28	29	30	31							
32	33	34	35	36	37							
51	52	53	54	55	56							
57	58	59	60	61	62							
76	77	78	79	80	81							
82	83	84	85	86	87							
O	N	M	L	I	H	G	F	E	D	C	B	A
13	14	15	16	17	18	19	20	21	22	23	24	25
38	39	40	41	42	43	44	45	46	47	48	49	50
63	64	65	66	67	68	69	70	71	72	73	74	75
88	89	90	91	92	93	94	95	96	97	98	99	00

Figura 1.3: Tabella della cifratura di Tritemio presa da [19].

Vediamo, in pratica, in che cosa consisteva esattamente l'idea di Tritemio. Supponiamo, per un istante, di voler mandare un messaggio ad un nostro amico Tizio, avvertendolo di non fidarsi di un certo Caio. Abbiamo, però, paura che Caio legga la nostra corrispondenza e, quindi, usando la Steganografia, scriviamo il seguente testo:

"Nelle ore notturne feroci illusioni di antichi riti tramandati in dimenticate isole, ci assalgono, ivi ora..."

Tizio, per leggere il messaggio originale che gli volevamo mandare, non dovrà far altro che leggere tutte le iniziali delle parole e comporre il testo:

"Non fidarti di Caio..."

Questo è l'esempio più semplice di tutti gli schemi proposti da Tritemio, il quale elaborò 40 sistemi principali e 10 sottosistemi secondari, sfruttando non solo le varie combinazioni di acronimi, ma anche usando dei dischi rotanti basati sulla sostituzione mono - alfabetica di Cesare. L'idea fondamentale di Tritemio era, quindi, quella di nascondere un testo segreto dentro un messaggio che funzionasse come copertura senza ricorrere a brutali sistemi fisici come, ad esempio, la rasatura dei capelli, ma sfruttando, al contrario, abili artifizi matematici e letterari. A meno di non sapere il sistema usato per nascondere il testo,

era, dunque, praticamente impossibile riuscire ad estrarre il significato reale del messaggio. L'unico inconveniente di tale metodo risiedeva nel fatto che il mittente ed il destinatario dovevano entrambi avere il libro di Tritemio per poter conoscere il sistema steganografico usato.

Le cifre nulle

La tecnica delle cifre nulle è una tecnica che consiste nel nascondere un messaggio segreto all'interno di un altro messaggio testuale. E' una tecnica molto utilizzata nella Seconda Guerra Mondiale soprattutto per comunicazioni via radio. I messaggi trasmessi venivano registrati e poi filtrati in modo tale da potervi estrarre il messaggio nascosto. Lo stratagemma che, in genere, veniva utilizzato era quello di considerare, ad esempio, le iniziali oppure la seconda lettera di ogni parola in sequenza al fine di risalire al messaggio nascosto. Ad esempio, il testo, riportato di sotto, è un messaggio che è stato mandato realmente durante la Seconda Guerra Mondiale da una spia tedesca:

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

Se consideriamo la seconda lettera di ogni parola, otteniamo il seguente messaggio nascosto:

"Pershing sails from NY June 1"

NOTA: in realtà, c'è una "r" di troppo e la "i" alla fine viene interpretata come "1".

Micropunti monografici

La tecnica dei micropunti monografici fu inventata dal direttore dell' F.B.I. durante la Seconda Guerra Mondiale. Si tratta di fotografie della dimensione di un punto dattiloscritto che, una volta sviluppate ed ingrandite, possono diventare pagine stampate di buona qualità.

Shakespeare e Bacon

Un esempio di Steganografia molto interessante è rappresentata nelle opere di Shakespeare. Secondo molti studiosi, infatti, molte opere dello scrittore inglese possono essere attribuite al noto scrittore e statista Francis Bacon. Questo perché all'interno di queste opere vi sono diversi testi nascosti che contengono il nome di Bacon stesso. A rafforzare questa ipotesi contribuiscono interessanti retroscena che accomunano Shakespeare a Bacon.

Al-Qaida

Oggiorno la Steganografia è anche utilizzata nell'ambito politico-militare. A tale proposito, il governo degli USA ha affermato che Osama bin Laden e l'organizzazione al-Qaida abbiano usato la Steganografia per inviare messaggi attraverso dei siti internet. Infatti, un famoso quotidiano americano "USA Today" del 10 Luglio 2002 riporta la seguente notizia:

"Ultimamente al-Qaida ha inviato centinaia di messaggi crittografati, nascosti in fotografie digitali sul sito "eBay.com". Molti dei messaggi sono stati inviati da caffè pakistani e librerie pubbliche di tutto il mondo..."

E, ancora:

"Ufficiali americani dicono che "azzam.com" contiene messaggi crittografati nelle sue immagini e nei suoi testi (pratica conosciuta come Steganografia). Essi affermano che i messaggi contengono istruzioni per i nuovi attacchi di al-Qaida."

Tuttavia, fino ad ora, non è stata trovata alcuna prova sostanziale a sostegno di queste affermazioni, quindi, o al-Qaida ha creato ed usato algoritmi steganografici davvero buoni, oppure si tratta di un'affermazione probabilmente falsa.

1.3.2 La steganografia nell'arte

Anche in ambito artistico è possibile individuare alcune opere, apparentemente "innocue", che contengono "qualcosa di strano", se osservate attentamente. In seguito, sono riportate solamente alcune di queste.³

Giotto

Nella Basilica di San Francesco d'Assisi (Padova) si trovano molte opere di Cimabue e Giotto. A quest'ultimo venne affidato il compito di rappresentare la vita di San Francesco attraverso vari affreschi verso gli anni 1280-1290. Solo in seguito a vari restauri avvenuti recentemente su un affresco (5 Novembre 2011), è stato trovato il viso di un demone nascosto in una nube.



Figura 1.4: Uno degli affreschi di Giotto rappresentante la vita di San Francesco preso da [2].

La cosa strana è che questo affresco è stato lì per tantissimi anni, guardato da milioni di persone e mai nessuno aveva notato quel "piccolo" dettaglio. Non si conosce ancora il vero motivo per cui egli abbia dipinto un demone nella nube più vicina all'angelo di destra. Tuttavia, questo si potrebbe ricondurre alle credenze che vi erano durante il Medioevo; infatti, si riteneva che nel cielo abitassero i demoni, che ostacolavano la salita delle anime verso il Paradiso.

Antonio Vassillachi

Nell'Abbazia di San Pietro a Perugia, edificata intorno al 996, sul muro di fronte all'altare maggiore, si trova un quadro del 1252 di Antonio Vassillachi. Questa enorme tela prende il nome di "Trionfo dell'ordine dei Benedettini", in quanto raffigura la celebrazione di San Benedetto e dell'ordine dei Benedettini con al centro raffigurato proprio il santo, celebrato da papi e vescovi in una sorta di convivio. Ma se si osserva questo dipinto da lontano,

³il seguente paragrafo fa riferimento a [2].

dall'altare maggiore, si può notare che tutte le figure formano un lugubre teschio con occhi infuocati, in cui il santo diventa il setto nasale e gli astri gli occhi di questo demone. E' evidente che ciò era voluto dall'artista, probabilmente un messaggio nascosto di quella che era la sua visione della gerarchia ecclesiastica: come a dire che la Chiesa era vittima di una perversione demoniaca.



Figura 1.5: "Trionfo dell'ordine dei Benedettini" di Antonio Vassillachi preso da [2].

Leonardo Da Vinci

Come è ben evidente, Leonardo Da Vinci ha voluto nascondere senz'altro dei messaggi segreti all'interno delle sue opere. "L'ultima cena" insieme a "La Gioconda" sono, senza ombra di dubbio, le opere più emblematiche del Rinascimento italiano. Nella prima opera, probabilmente commissionata dal duca Ludovico Sforza di Milano per il refettorio di Santa Maria delle Grazie, si suppone che l'artista abbia voluto rappresentare secondo quanto riportato nei Vangeli gnostici.



Figura 1.6: "L'ultima cena" di Leonardo Da Vinci preso da [2].

L'apostolo alla destra di Gesù non sarebbe Giovanni, bensì Maria Maddalena, come il grande spazio che forma una "V" tra i due vorrebbe far intendere. Questo noto simbolo sta per la sacralità della donna, del femminismo sacro. Infatti, vorrebbe simboleggiare un calice, con cui da sempre viene stilizzato il sesso femminile. Inoltre, se si tracciano con delle rette i contorni di queste due figure, si ottiene una "M", l'iniziale di Maria Maddalena, appunto.



Figura 1.7: Alcuni dettagli che emergono dall'opera "L'ultima Cena" di Leonardo Da Vinci preso da [2].

⁴Certamente, "La Gioconda" rimane tutt'ora l'opera più emblematica di Leonardo Da Vinci e quella che ha sempre fatto discutere, alimentando la fantasia di molti scrittori e romanzieri. Basti pensare al famoso best seller "Il Codice Da Vinci" di Dan Brown.

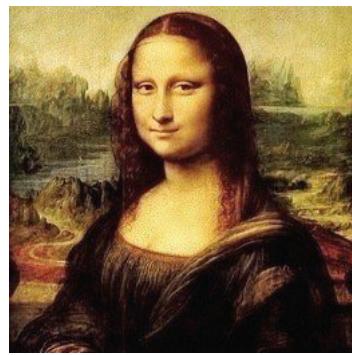


Figura 1.8: "La Gioconda" di Leonardo Da Vinci preso da [8].

Ma se, in quest'ultimo, la Mona Lisa era la chiave per trovare la tomba di Maria Maddalena ed il Santo Graal, nell'ultimo lavoro della storica dell'arte Carla Glori ("Enigma Leonardo: la Gioconda, in memoria di Bianca.") il dipinto si trasforma in una sorta di cartina geografica che permette di arrivare fino all'identità della misteriosa donna ritratta. Infatti, secondo la storica, il paesaggio, ritratto alle spalle della donna, altro non è che la valle di Bobbio ed il ponte è il cosiddetto Ponte del Diavolo o Ponte Gobbo sul Trebbia.



Figura 1.9: L'attuale ponte Gobbo sul Trebbia preso da [1].

Il ponte è già di per sé misterioso. Infatti, la leggenda vuole che esso sia stato costruito dal demonio (da qui il nome): Satana lo realizzò in una sola notte, dopo aver stipulato un patto con San Colombano, che gli promise in cambio l'anima del primo viaggiatore che lo avrebbe percorso. Ma il santo si beffò del diavolo, facendovi passare sopra un cagnolino. Ritornando a "La Gioconda", la studiosa è convinta che il ponte ritratto sia proprio quello sul Trebbia ed il paesaggio quello della campagna intorno a Bobbio. Questa

⁴Per maggiori dettagli e per le immagini fare riferimento a [8] e [1].

è un'ipotesi che porterebbe ad identificare e, quindi, a dare un nome alla donna ritratta da Leonardo: Bianca Giovanna Sforza, figlia di Ludovico il Moro, signore di Bobbio. Inoltre, lo storico dell'arte Silvano Vincenti, presidente del "Comitato Nazionale per la Valorizzazione dei Beni Storici ed Ambientali", avrebbe individuato negli occhi della donna ritratta due lettere: una "G" ed una "S", sebbene la questione sia ancora dibattuta. Nelle ipotesi di Vincenti, però, le due iniziali sarebbero le iniziali di Giovanna Sforza.

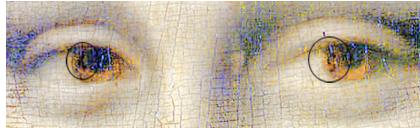


Figura 1.10: Dettaglio degli occhi de "La Gioconda" preso da [8].

Inoltre, Vincenti sostiene che, sempre sotto le arcate del ponte, si trovino i numeri "7" e "2", che lo studioso ricollega all'anno 1472, quando il ponte venne distrutto dalla piena del Trebbia.

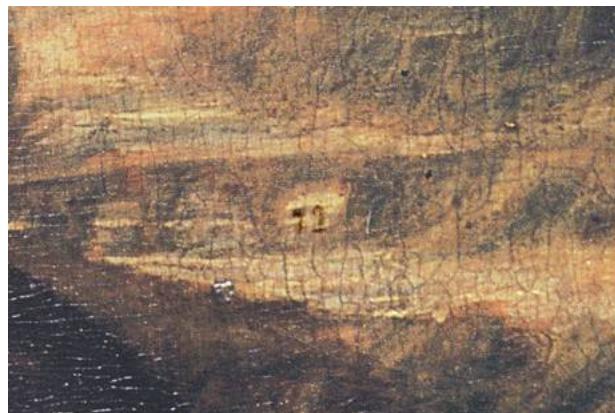


Figura 1.11: Dettagli dell'arcata del ponte, che si presume rappresentino i numeri "7" e "2" preso da [1].

La studiosa ritiene che si tratti, quindi, proprio di quel ponte, dal momento che nel quadro ci sarebbe anche una strada serpentina tuttora visibile a Bobbio. Tuttavia, ci sono molti battibecchi nel mondo dell'arte riguardo a queste ipotesi. Una di queste è quella del giornalista Massimo Polidoro, il quale fa notare la mancanza di qualsiasi documento che attesti che Leonardo abbia fatto visita a Bobbio, che sarebbe dovuta avvenire prima del 1472, anno della distruzione del ponte. Questo fatto è ritenuto improbabile, dal momento che prima di quella data Leonardo era un ragazzo apprendista nella bottega del Verrocchio. Inoltre, per quanto riguarda le iniziali individuate negli occhi della donna, egli ricorda che precedentemente erano state interpretate "CE" o "CB".

Inoltre, anche l'identità della donna non è tuttora chiara: c'è chi crede si tratti di Lisa Gherardini, moglie del mercante fiorentino Francesco del Giocondo, c'è chi, al contrario, ritiene sia Bianca Giovanna Sforza, figlia di Ludovico il Moro.

Ma sono proprio i due storici d'arte Carla Glori e Silvano Vincenti a confermare la seconda ipotesi.

1.3.3 Terminologia utilizzata

⁵Prima di addentrarci nei dettagli della Steganografia, è bene fornire alcune definizioni che ci torneranno utili nel corso della seguente trattazione.

Abbiamo, in precedenza, definito la Steganografia come l'arte della "scrittura coperta". Ora, affinché possa esserci una comunicazione segreta, è necessario individuare il "file contenitore", detto "cover medium", il quale, appunto, dovrà contenere l'informazione steganografica, che il mittente invierà successivamente al destinatario. Per "informazione steganografica" si intende il messaggio segreto che il mittente vorrebbe inviare al destinatario. Per quanto riguarda il "cover medium", invece, esso può essere un qualunque file multimediale come, ad esempio, un file testuale, un'immagine digitale, una traccia audio, un video, ecc... Il processo che incorpora ("embedded") l'informazione steganografica all'interno di un oggetto contenitore, viene detto "processo di embedding", alla fine del quale si ottiene il cosiddetto "stego medium". Affinché lo "stego medium" non desti sospetti agli occhi di un eventuale osservatore, o meglio "nemico", è necessario che questo sia quasi del tutto simile al nostro "cover medium". Una volta ottenuto lo "stego medium", questo può essere inviato al destinatario, il quale, a partire da esso ed applicando un opportuno algoritmo di estrazione, sarà in grado di estrarre l'informazione steganografica contenuta al suo interno. Recapitolando, quindi, un sistema steganografico è costituito essenzialmente da due processi: un processo di embedding, il cui obiettivo è quello di inserire l'informazione steganografica a partire da un "cover medium" e da un messaggio segreto, ed un processo di estrazione, il cui obiettivo principale è quello di estrarre correttamente l'informazione nascosta a partire da uno "stego medium".

Possiamo schematizzare i due processi, come riportato nella seguente figura:

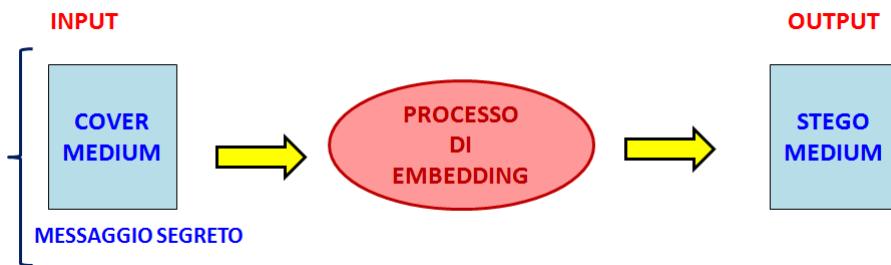


Figura 1.12: Schema di base del processo di embedding di un sistema steganografico.



Figura 1.13: Schema di base del processo di estrazione di un sistema steganografico.

⁵Questo paragrafo fa riferimento a [21].

1.3.4 Il problema del prigioniero

⁶Lo studio della Steganografia è spesso descritto attraverso il "Problema del Prigioniero", formulato da Simmons nel 1983. Alice e Bob sono due prigionieri che devono escogitare un piano per poter fuggire. Essi si scambiano dei messaggi attraverso il guardiano Wendy. Se Wendy scopre che i due si stanno scambiando dei messaggi, quindi, in pratica, se il guardiano è attivo (in caso contrario, il guardiano si dirà passivo), metterà uno di loro in isolamento ed il piano fallirà. Quindi, devono trovare un metodo per nascondere il loro testo cifrato in un testo apparentemente innocuo. Tuttavia, il problema del prigioniero è solo un modello che mira semplicemente a spiegare i concetti della Steganografia: infatti, non ci fornisce nessun dettaglio su come realizzare il tutto.

1.3.5 Proprietà di un sistema steganografico

⁷Affinchè ci possa essere una comunicazione nascosta, è necessario scegliere con cura il "cover medium", il quale, in seguito al processo di embedding, avrà il compito di mantenere nascosta l'informazione steganografica. Un buon sistema steganografico deve presentare essenzialmente 4 requisiti fondamentali:

1. Capacità di incorporamento

La capacità di incorporamento è la quantità di dati che può essere nascosta all'interno di un "cover medium". Pertanto, prima di effettuare l'embedding dei dati, è necessario verificare che la dimensione del "cover medium" sia maggiore rispetto alla dimensione dell'informazione steganografica da nascondere al suo interno. Infatti, se la dimensione del messaggio segreto risultasse essere maggiore rispetto alla dimensione del "cover medium", allora l'inserimento non può essere effettuato. Ovviamente, si vorrebbe avere a disposizione un "cover medium" che permetta di incorporare al suo interno quanti più dati possibili.

2. Impercettibilità

L'impercettibilità si riferisce al fatto che lo "stego medium" debba essere il più simile possibile al "cover medium", in modo tale da non insospettire un eventuale osservatore. Infatti, se lo "stego medium" suscita dei sospetti, allora la Steganografia fallisce.

3. Sicurezza

La Steganografia potrebbe essere affetta da attacchi passivi o attivi, se volessimo parlare nei termini del "Problema del Prigioniero" alludendo al guardiano Wendy. Quindi, in pratica, si vorrebbe avere una comunicazione che, in un certo senso, sia "invisibile" agli occhi di persone che non hanno alcuna connessione tra mittente e destinatario.

4. Resistenza alla manomissione

Si intende la resistenza al malfunzionamento o al sabotaggio da parte di utenti che non possono avere accesso alle informazioni nascoste nello "stego medium".

NOTA: E' bene tener presente che tutti i requisiti non possono essere soddisfatti contemporaneamente. Infatti, ad esempio, una maggiore capacità di incorporamento, porterà a ridurre senza ombra di dubbio la sicurezza e la resistenza alla manomissione.

⁶Il seguente paragrafo fa riferimento a [7] e [19].

⁷Il seguente paragrafo fa riferimento a [3].

1.3.6 Classificazione delle tecniche steganografiche

⁸Nella letteratura ci sono due classificazioni delle tecniche steganografiche. La prima classificazione suddivide le tecniche steganografiche in

1. Steganografia Iniettiva

E' la tecnica steganografica più utilizzata e consiste nell'"iniettare" il messaggio segreto all'interno di un messaggio contenitore già esistente, modificandolo in modo tale da contenere il messaggio e da risultare agli occhi umani praticamente indistinguibile dall'originale.



Figura 1.14: Steganografia iniettiva presa da [19].

2. Steganografia Generativa

E' una tecnica che, a partire dal messaggio segreto, è in grado di generare dei potenziali messaggi contenitori, atti a nascondere nel migliore dei modi il messaggio segreto.



Figura 1.15: Steganografia generativa [19].

La seconda, invece, le classifica in

1. Steganografia Sostitutiva

E' la tecnica steganografica più utilizzata e la sua idea di base è la seguente: la maggior parte dei canali di comunicazione (come, ad esempio, linee telefoniche, un'immagine scannerizzata, trasmissioni radiofoniche, ecc...) trasmettono dei segnali che sono accompagnati quasi sempre da un rumore. Questo rumore può essere sostituito da un segnale (il messaggio segreto), che può essere trasformato, a meno di conoscere la chiave segreta, in un rumore indistinguibile dal rumore vero e proprio (almeno dall'occhio umano) e, quindi, può essere trasmesso senza destare dei sospetti. In pratica, consiste nel sostituire i bit meno significativi (LSB - "Less Significant Bit") del rumore presente nei file digitali con i bit del messaggio segreto. Una volta fatto ciò, il file contenitore risultante risulta essere del tutto e per tutto simile all'originale con differenze difficilmente percettibili. Pertanto, a meno di confronti approfonditi con il file originale, è difficile dire se le eventuali perdite di qualità siano dovute al rumore oppure alla presenza di messaggi nascosti. In molti casi, però, il file originale non è disponibile, quindi, questo confronto è impossibile. L'unico svantaggio che presenta questa tecnica è che le alterazioni possono alterare le caratteristiche statistiche del rumore del media utilizzato. Infatti, se il "nemico" possiede il modello del rumore,

⁸Questo paragrafo fa riferimento a [19].

può utilizzarlo per testare se il file contiene un eventuale messaggio steganografico. Il problema di questo attacco, tuttavia, sta nella difficoltà di costruire un modello che tenga conto di tutti i possibili errori o rumori. In molti casi specifici questo attacco ha comunque ottenuto dei risultati positivi. Di seguito, sono illustrate due tecniche steganografiche in grado di evitare questo tipo di attacco.

2. Steganografia Selettiva

La Steganografia Selettiva ha un valore puramente teorico, in quanto dal punto di vista pratico non viene utilizzata. L'idea di base di questa tecnica è procedere per tentativi, ripetendo uno stesso procedimento fino a quando il risultato non soddisfi una certa condizione. Viene, in pratica, impostata una **funzione hash** molto semplice, il cui compito è quello di verificare la parità dei bit di un'immagine digitale:

- se l'immagine cover contiene un numero di bit pari ad "1" dispari, assume valore 1;
- se l'immagine cover contiene un numero di bit pari a "0" pari, assume valore 0.

Quindi, praticamente, quando si acquisisce il file binario, si controlla se il numero di bit pari ad "1" oppure a "0" sia rispettivamente dispari oppure pari. Nel caso in cui sia pari, allora vuol dire che abbiamo trovato un file adatto a contenere la nostra informazione steganografica, altrimenti si procede con l'acquisizione di un altro file digitale. Un vantaggio di questa tecnica è che non si può dire in alcun modo se le caratteristiche del rumore presentino una distorsione rispetto ad un modello di riferimento. Tuttavia, lo svantaggio principale sta nel fatto che risulta essere molto dispendiosa rispetto alla scarsa quantità di informazioni che può contenere. Per questo motivo nella pratica non viene utilizzata.

3. Steganografia Costruttiva

Anche questa tecnica steganografica, come quella selettiva, tenta di sostituire il rumore presente nel media, utilizzato come cover, con l'informazione segreta opportunamente modificata, in modo tale da imitare le caratteristiche statistiche del rumore originale. Secondo questa tecnica, 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 l'informazione steganografica, sia il più possibile conforme al modello. Questo approccio è senza dubbio valido, ma presenta anche alcuni svantaggi. Prima di tutto, 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 e di risorse migliori, riesca a costruire un modello più accurato, riuscendo ancora a distinguere tra il rumore originale ed uno sostituto. Inoltre, se il modello del rumore, utilizzato dal metodo steganografico, dovesse cadere nelle mani di un eventuale "nemico", egli lo potrebbe analizzare 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 lo stesso sistema.

1.4 Attacco alla Steganografia: la Steganalisi

⁹La comunicazione steganografica fallisce nel momento in cui un eventuale osservatore sospetti che nello "stego medium" ci sia qualcosa di "anomalo", pur non estraendo di fatto l'informazione steganografica in esso contenuta. La Steganalisi si occupa di attaccare i Metodi Steganografici attraverso il rilevamento, l'estrazione, la distruzione e la manipolazione dei dati nascosti in uno "stego medium". Un buon steganalista deve essere consapevole dei metodi e delle tecniche steganografiche per poter attaccare in modo efficace un sistema steganografico. A seconda delle informazioni in possesso da parte di un presunto "attaccante", possiamo classificare gli attacchi alla Steganografia nel seguente modo:

- **Stego only attack:** per l'analisi è disponibile solamente lo "stego medium".
- **Known cover attack:** è noto sia lo "stego medium", sia il "cover medium".
- **Known message attack:** in alcuni casi, è possibile che si conosca il messaggio segreto.
- **Chosen stego attack:** sono noti lo "stego medium" e l'algoritmo steganografico utilizzato.
- **Chosen message attack:** la Steganalisi spesso crea alcuni "stego medium" come esempi da utilizzare per alcuni sistemi steganografici. In seguito, si analizzano questi "stego medium" con degli oggetti che si presume siano degli "stego medium" e si cerca di risalire all'algoritmo steganografico utilizzato.
- **Known stego attack:** sono noti il "cover medium" e l'algoritmo steganografico utilizzato.

1.4.1 Principio di Kerckhoffs

¹⁰Un buon sistema steganografico deve essere, pertanto, "imbattibile", se sottoposto a qualsiasi attacco di Steganalisi. Una buona idea sarebbe quella di combinare la Steganografia con la Crittografia, in quanto, anche se un eventuale "nemico" riuscisse a risalire al sistema steganografico utilizzato e, quindi, ad estrarre l'informazione steganografica presente al suo interno, quest'ultima si presenterebbe del tutto incomprensibile. Quindi, in pratica, prima del Processo di Embedding, sarebbe utile cifrare il messaggio nascosto, servendosi di una chiave segreta (che deve rimanere tale), e successivamente provvedere ad inserire l'informazione steganografica, a questo punto crittografata, all'interno del "cover medium". Questo, come è evidente, aumenterebbe il livello di sicurezza che, come sappiamo, è una delle proprietà più importanti di un buon sistema steganografico.

Questa soluzione, in realtà, è stata illustrata nel famoso "**Principio di Kerckhoffs**" nell'ambito della Crittografia (1883). Kerckhoffs ritiene, infatti, che "la sicurezza del crittosistema deve basarsi sull'ipotesi che il nemico abbia piena conoscenza dei dettagli di progetto ed implementazione del crittoalgoritmo; la sola informazione di cui il nemico non deve disporre è una sequenza di numeri casuali che costituisce la *chiave segreta* senza la quale, osservando un canale di comunicazione, non deve avere neanche la più piccola possibilità di verificare che è in corso una comunicazione segreta."

⁹Questo paragrafo fa riferimento a [21].

¹⁰Si fa riferimento a [19].

PROCESSO DI EMBEDDING



PROCESSO DI ESTRAZIONE



Figura 1.16: Schema di Embedding e di Estrazione di Steganografia combinata alla Crittografia.

1.5 Steganografia vs Crittografia

¹¹Sia la Crittografia che la Steganografia hanno come obiettivo comune quello di garantire una comunicazione segreta. Però, la fanno in maniera differente:

- la Crittografia rende incomprensibile il messaggio agli occhi di un eventuale nemico; quindi, in pratica, il messaggio è ben visibile. Esso può essere decifrato se e solo se si è in possesso della corretta chiave segreta, utilizzata per crittografare il messaggio segreto. Pertanto, la comunicazione crittografica fallisce nel momento in cui il "nemico" è in possesso della chiave segreta e riesce, di conseguenza, ad ottenere il messaggio in chiaro.
- La Steganografia, al contrario, ha come obiettivo principale quello di nascondere l'esistenza stessa del messaggio segreto; quindi, in pratica, il messaggio è come se fosse del tutto invisibile agli occhi di un eventuale "nemico". Nel momento in cui un eventuale osservatore sospetta che all'interno dello "stego medium" possa esserci un eventuale messaggio nascosto, pur non estraendovi l'informazione steganografica, la comunicazione steganografica fallisce.

¹¹Si fa riferimento a [21] e [4].

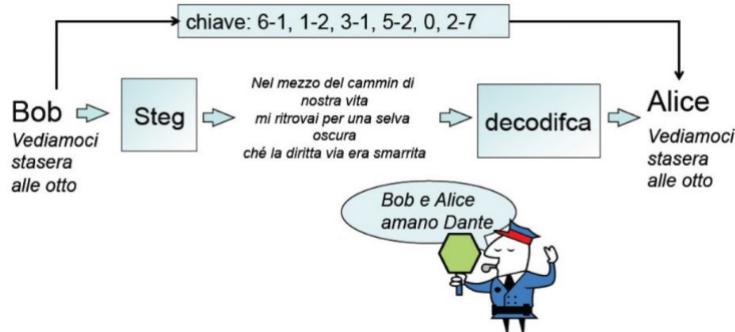
Possiamo riassumere le differenze tra Crittografia e Steganografia nella seguente tabella:

Differenze tra Steganografia e Crittografia

Steganografia	Crittografia
<ul style="list-style-type: none"> - I messaggi sono "invisibili". - La Steganografia ha come obiettivo principale quello di mantenere nascosta l'esistenza stessa della comunicazione. - Scienza poco conosciuta. - Tecnologia ancora in fase di sviluppo per alcuni formati di file digitali. - Una volta rilevato il messaggio, è noto. - La Steganografia non altera la struttura del messaggio segreto. 	<ul style="list-style-type: none"> - I messaggi sono "visibili". - La Crittografia ha come obiettivo quello di impedire ad eventuali osservatori di comprendere il contenuto del messaggio. - Una tecnologia molto utilizzata. - La maggior parte degli algoritmi sono noti a tutti. - Gli algoritmi sono, in genere, resistenti ad eventuali attacchi. Infatti, è necessaria un potenza di calcolo non del tutto indifferente. - La Crittografia altera la struttura del messaggio segreto.

Di sotto è fornito un esempio "pratico" di differenza tra Steganografia e Crittografia.

STEGANOGRAFIA



CRITTOGRAFIA

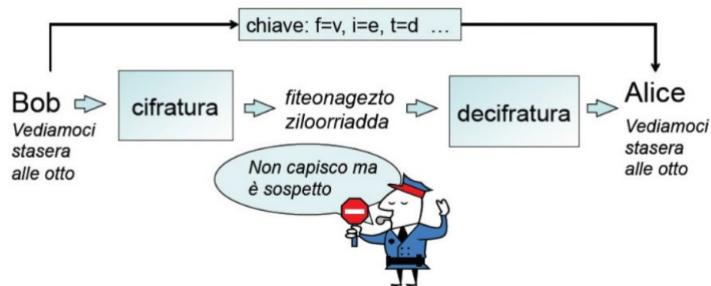


Figura 1.17: Steganografia vs. Crittografia preso da [17].

1.6 Le motivazioni

La Steganografia ha molte applicazioni pratiche soprattutto riguardanti la protezione del copyright e l'autenticazione ed il monitoraggio e tracciamento dei dati trasmessi.

- **Protezione del copyright e autenticazione**

In questo particolare contesto, possiamo pensare alla Steganografia come ad una sorta di **Watermarking digitale "invisibile"**¹². In pratica, è possibile incorporare all'interno di un file digitale, come ad esempio un'immagine digitale, una "filigrana invisibile" (contenente, ad esempio, la firma di colui/colei che ha scattato una fotografia), in modo tale da poterne rilevare eventuali modifiche effettuate in seguito (ad esempio, il ritaglio di una foto, l'aggiunta dei filtri, ecc...). Un'altra applicazione pratica della protezione del copyright potrebbe essere, ad esempio, quella che riguarda i **referti medici** di un paziente, al fine di evitare di rilevarne l'identità per questioni di privacy.

¹²Il termine "watermark" significa letteralmente "filigrana". In pratica, si tratta di un'informazione inamovibile, solida e quasi del tutto impercettibile. Di solito, viene incorporata nei dati ed è tale da non poter essere rimossa. In genere, contiene informazioni sull'origine, sullo stato/destinazione dei dati e consente delle appropriate azioni di follow-up in caso di sospette violazioni del copyright. Nel caso del Watermarking digitale, il watermark è ben "visibile"; mentre nella Steganografia è del tutto "invisibile" agli occhi di un eventuale osservatore. Per ulteriori dettagli sul **Watermarking digitale** e sulle sue tecniche, fare riferimento a [10].

- **Monitoraggio e tracciamento dei dati trasmessi**

Un esempio di monitoraggio e tracciamento dei dati trasmessi potrebbe essere quello della registrazione e del monitoraggio automatico dei programmi radiofonici e televisivi, per consentirne il pagamento automatico dei canoni ai legittimi proprietari e garantire, quindi, che non ci siano violazioni dei diritti di proprietà intellettuale dei dati trasmessi.

1.7 I contributi

Come è già stato accennato in precedenza, il nostro obiettivo principale sarà quello di fornire una nuova tecnica steganografica che ci permetta di inserire e di estrarre informazioni segrete, utilizzando il formato PDF ("Portable Document Format"), per questioni strettamente legate alla privacy (per esempio, nel caso dei referti medici) oppure al copyright. Ci occuperemo essenzialmente di Steganografia sul testo, che per molti versi è meno utilizzata, in quanto modificare un file testuale senza che le modifiche risultino essere impercettibili, risulta essere alquanto difficile, ma anche di Steganografia sulle immagini, utilizzando l'algoritmo F5, di cui verrà fornita una trattazione dettagliata in seguito. Pertanto, quello che faremo sarà:

- utilizzare come cover dei file PDF, che possono contenere solo testo oppure sia testo che immagini;
- se il PDF contiene solamente del testo, verrà utilizzata una nuova tecnica steganografia sul testo;
- se il PDF contiene sia immagini sia testo, verrà combinata la tecnica steganografica sul testo con quella sulle immagini, facendo uso dell'algoritmo F5.

Per poter implementare un processo steganografico di questo tipo, è stata utilizzata la libreria **iText** per Java, che fornisce una serie di metodi per la lettura, la scrittura e la modifica di file PDF. Il **codice sorgente Java**, relativo all'applicazione sviluppata per un processo steganografico di questo tipo, è disponibile su <https://github.com/roccchinaRomano?tab=repositories>, nella repository “**steganografiaSuPDF**”, all'interno della quale è possibile trovare il file “**README.txt**”, nel quale vi sono tutte le informazioni necessarie per il download e l'utilizzo dell'applicazione.

1.8 Struttura della Tesi

La Tesi presenta essenzialmente la seguente struttura:

- il **Capitolo 2** fornisce una panoramica generale sulle attuali tecniche steganografiche esistenti sul testo e sulle immagini (si soffermerà, in particolare, sull'algoritmo F5) e sui relativi attacchi di steganalisi;
- il **Capitolo 3** introduce una nuova tecnica steganografica sul testo, in particolare, utilizzando il formato PDF come *cover*, di cui verrà fornita una descrizione, evidenziandone soprattutto i vantaggi. Dopodiché ci soffermeremo sugli aspetti implementativi: in particolare, la libreria *iText*, che si occupa di fornire dei metodi per la lettura, scrittura e modifica dei file PDF. Inoltre, saranno forniti degli esempi di

utilizzo dell'applicazione sviluppata (esempi pratici che mostreranno come avviene il processo di embedding e di estrazione di un'informazione steganografica a partire da un file PDF).

- il **Capitolo 4** mostrerà alcuni risultati sperimentali, ottenuti utilizzando l'algoritmo proposto in questa tesi.
- il **Capitolo 5** riassume l'intero lavoro svolto e propone alcuni sviluppi futuri.

Capitolo 2

STATO DELL'ARTE DELLE TECNICHE STEGANOGRAFICHE

Il seguente capitolo ha l'obiettivo di fornire una panoramica in merito alle attuali Tecniche Steganografiche ed ai relativi attacchi di Steganalisi, cui possono essere esposte. In particolare, per i fini della nostra trattazione, ci soffermeremo solamente sulle Tecniche Steganografiche che hanno come "cover medium" il testo e le immagini.

2.1 La Steganografia sulle Immagini

¹ La Steganografia sulle immagini digitali è molto utilizzata, in quanto le immagini presentano al loro interno molte informazioni ridondanti, che possono essere utili ai fini della Steganografia, ed, in più, la loro implementazione risulta essere molto "semplice". La Steganografia sulle immagini utilizza come "cover medium" e "stego medium" un'immagine per incorporare le informazioni steganografiche; pertanto, parleremo rispettivamente di "*cover image*" e di "*stego image*". Per poter nascondere messaggi segreti all'interno delle immagini digitali, è necessario avere a disposizione tecniche che siano in grado di far apparire lo "*stego image*" visivamente del tutto simile alla "*cover image*". Tuttavia, per le immagini digitali esistono molti formati (come, ad esempio, i formati ".png", ".jpeg", ".bmp", ".gif", ecc...).

Possiamo classificare le Tecniche Steganografiche sulle immagini in 3 principali categorie: *la Tecnica del bit meno significativo (LSB)*, *Mascheramento e filtraggio*, *Algoritmi e trasformazioni*.

2.1.1 Tecnica di inserimento del bit meno significativo(LSB)

La **Tecnica LSB** ("Least Significant Bit"), o più comunemente nota come la **Tecnica del bit meno significativo**, è la tecnica steganografica più utilizzata per le immagini, perché la sua implementazione risulta essere molto facile. In pratica, vengono sostituiti i bit meno significativi dei pixel delle immagini con i bit dell'informazione steganografica da inserire. Pertanto, è necessario utilizzare delle immagini, aventi un formato la cui compressione è

¹Questo paragrafo fa riferimento principalmente a [16], [21], [20], [25] e [15].

senza perdita² (come, ad esempio, il formato PNG), altrimenti, in caso contrario (cioè nel caso in cui facessimo uso di un formato la cui compressione è con perdita³ di informazioni come, ad esempio, il formato JPEG), le informazioni nascoste potrebbero andar perse. Ciascun pixel di un'immagine digitale, in genere, può essere rappresentato attraverso dei valori numerici, che, ad esempio, indicano il colore oppure l'intensità.

In generale, le immagini possono essere divise in 2 gruppi:

- **Immagini a 8 bit**

In immagini di questo tipo può essere nascosta solamente 1 bit di informazione per ciascun pixel dell'immagine. Dal momento che le immagini a 8 bit possono avere un numero massimo di 256 colori, esse richiedono particolare attenzione: infatti, effettuando una sostituzione LSB, potrebbe verificarsi una variazione di colore visibile ad occhio nudo. Per questo motivo, bisogna prestare attenzione quando si sceglie una "cover image". Una soluzione a questo problema potrebbe essere quella di utilizzare un'immagine a scala di grigi (per intenderci, un'immagine grigia); infatti, in questo caso, la differenza di colore tra i pixel adiacenti, dovuta alla sostituzione LSB, risulterà essere meno evidente almeno visivamente.

- **Immagini a 24 bit**

Per un'immagine a 24 bit si ha che ogni pixel dell'immagine è di 24 bit. Ora, i 24 bit di ciascun pixel vengono suddivisi in 3 parti⁴, ciascuna delle quali è costituita da 8 bit. Il vantaggio di queste immagini consiste nel fatto che è possibile incorporare 3 bit di informazione per ciascun pixel: in pratica, 1 bit per gli 8 bit del colore rosso, 1 bit per gli 8 bit del colore blu ed 1 bit per gli 8 bit del colore verde. Quindi, a differenza delle immagini a 8 bit, è possibile memorizzare una quantità di dati maggiore, evitando anche di ottenere delle variazioni di colore evidenti, dovute alla sostituzione LSB. Si ottiene, pertanto, un'immagine stego quasi del tutto simile all'immagine cover.

Tuttavia, pur essendo semplice l'implementazione, è una tecnica meno robusta, dal momento che i dati possono andar persi in seguito ad una manipolazione dell'immagine oppure a semplici attacchi.

Di sotto è riportato un esempio di 3 pixel di un'immagine a 24 bit. Ciascun pixel è costituito da 3 parti, ognuna dei quali risulta essere di 8 bit. Supponiamo di voler effettuare l'inserimento del carattere "A" in questi 3 pixel, il cui valore binario è 01000001, utilizzando l'inserimento LSB.

²oppure nota come **compressione dei dati lossless**.

³oppure nota come **compressione dati lossy**.

⁴Si fa riferimento alla suddivisione dell'immagine digitale in tre matrici, che indicano i colori rosso(RED), blu (BLUE) e verde(GREEN).

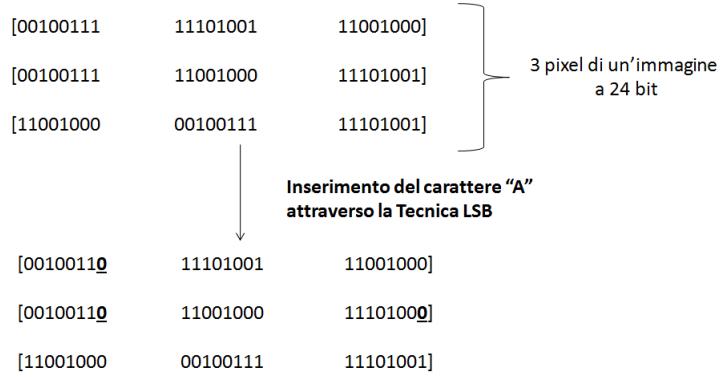


Figura 2.1: Un esempio di sostituzione LSB in un’immagine a 24 bit.

2.1.2 Mascheramento e filtraggio

Le tecniche di mascheramento e filtraggio, generalmente limitate ad immagini a 24 bit o a scala di grigi, adottano un approccio diverso, che consiste nel nascondere le informazioni in maniera del tutto simile al "watermarking" di carta. Quindi, le informazioni steganografiche sono incorporate in modo tale da diventare parte integrante delle immagini e, di conseguenza, i dati non andranno persi anche in caso di una compressione che prevede perdita di dati (come accade per le immagini JPEG). Inoltre, l’inserimento è fatto in modo tale che l’occhio umano non sia in grado di percepire eventuali anomalie. Per questo motivo è, senza dubbio, più robusta rispetto alla Tecnica LSB.

2.1.3 Algoritmi e trasformazioni

E’ un modo più complesso di nascondere le informazioni segrete all’interno di un’immagine. Consiste nell’incorporare i dati nei coefficienti di trasformazione di un’immagine, utilizzando, ad esempio, i coefficienti della Trasformata Discreta del Coseno (DCT - "Discrete Cosine Transform"). Infatti, se incorporiamo delle informazioni nel dominio spaziale (come nel caso della Tecnica LSB), si potrebbero avere delle perdite di dati, soprattutto, in seguito, a delle compressioni (come accade per le immagini JPEG, la cui compressione è con perdita di dati) oppure in seguito a dei ritagli. Per ovviare a problemi di questo tipo, le informazioni vengono incorporate nel *dominio della frequenza* (dominio in cui opera, appunto, la DCT). Ad esempio, nel caso della DCT, che opera su immagini JPEG, le informazioni sono nascoste nei bit meno significativi (LSB) dei coefficienti DCT dell’immagine JPEG.

Il formato JPEG

Il formato *JPEG* ("Joint Photographic Experts Group") memorizza i dati dell’immagine utilizzando una compressione *con perdita* ("lossy") come frequenza dei coefficienti quantizzati. La figura riportata di sotto (presa da [25]), mostra i passaggi eseguiti per la compressione.

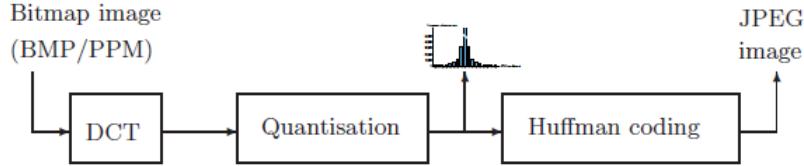


Figura 2.2: Processo di compressione del formato JPEG.

Per prima cosa, la compressione JPEG taglia l'immagine bitmap in gruppi di 8x8 pixel. Dopo aver calcolato la Trasformata Discreta del Coseno (DCT), la quantizzazione arrotonda opportunamente la frequenza dei coefficienti in numeri interi nell'intervallo (-2048, ..., 2047) ("fase con perdita"). L'istogramma, riportato di sotto (preso da [25]), mostra la distribuzione discreta della frequenza di occorrenza dei coefficienti.

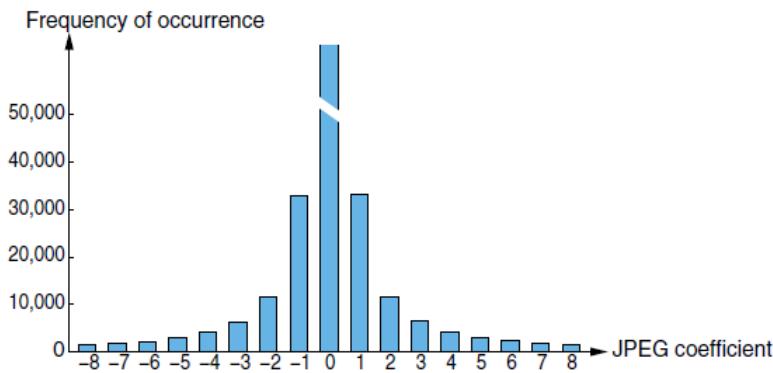


Figura 2.3: Istogramma dei coefficienti JPEG dopo la quantizzazione.

Se si osserva la distribuzione, è possibile individuare 2 proprietà:

1. la frequenza di occorrenza dei coefficienti diminuisce con l'aumentare del valore assoluto;
2. la diminuzione della frequenza di occorrenza dei coefficienti diminuisce all'aumentare del valore assoluto, cioè la differenza tra 2 barre dell'istogramma al centro è più grande rispetto quella che si trova al margine.

Dopo la quantizzazione con perdita, la *codifica di Huffman* garantisce la codifica senza ridondanza dei coefficienti quantizzati.

La Trasformata discreta del coseno (DCT)

La Trasformata Discreta del Coseno (DCT), utilizzata nell'algoritmo di compressione JPEG, trasforma blocchi successivi di 8x8 pixel dell'immagine in 64 coefficienti DCT ciascuno. Ogni coefficiente DCT $F(u, v)$ di un blocco di 8x8 pixel dell'immagine $f(x, y)$ è data dalla seguente relazione:

$$F(u, v) = \frac{1}{4}C(u)C(v) \left(\sum_{x=0}^7 \sum_{y=0}^7 f(x, y) * \frac{\cos(2x+1)u\pi}{16} \frac{\cos(2y+1)v\pi}{16} \right) \quad (2.1)$$

dove

$$C(x) = \begin{cases} \frac{1}{\sqrt{2}} & se \quad x = 0 \\ 1 & altrimenti \end{cases} \quad (2.2)$$

Dopo aver calcolato i coefficienti, viene eseguita la seguente operazione di quantizzazione (sempre sui coefficienti):

$$F^Q(u, v) = \left\lfloor \frac{F(u, v)}{Q(u, v)} \right\rfloor \quad (2.3)$$

dove $Q(u, v)$ è una tabella di quantizzazione di 64 elementi.

A questo punto, possiamo usare i bit meno significativi dei coefficienti DCT quantizzati come bit ridondanti per fare l'embedding del messaggio segreto. La modifica di un singolo coefficiente DCT influenzereà tutti i 64 pixel dell'immagine. Dal momento che le modifiche vengono effettuate nel dominio della frequenza anziché in quello spaziale, l'occhio umano non sarà in grado di rilevare eventuali modifiche apportate nell'immagine, dovute all'inserimento di un messaggio nascosto. Questo approccio è utilizzato nell'algoritmo F5, di cui parleremo dettagliatamente più avanti.

2.1.4 Tecniche di steganalisi sulle immagini

In seguito riportiamo una classificazione dei Metodi di Steganalisi utilizzati per rilevare eventuali anomalie nelle immagini, dovute all'incorporamento di informazioni steganografiche.

Attacco visivo

E' un tipo di attacco che può essere utilizzato per rilevare eventuali modifiche nell'immagine soprattutto se l'embedding dell'informazione steganografica è stata eseguita attraverso la Tecnica LSB. Quindi, in pratica, è possibile verificare e provare se esiste, almeno visivamente, una differenza, ad esempio, di colore tra i pixel adiacenti di una data immagine.

Attacco statistico

E' un attacco più difficile rispetto al precedente, anche se ha avuto successo soprattutto per il rilevamento di eventuali modifiche nelle immagini JPEG. L'approccio che si segue è quello di eseguire un'analisi statistica delle immagini mediante delle formule matematiche e, successivamente, rilevare la presenza di eventuali dati nascosti sulla base di questi risultati statistici. Generalmente, il messaggio nascosto occupa una posizione più casuale rispetto ai dati originali dell'immagine. Pertanto, se si riesce a trovare una formula matematica per conoscere proprio questa casualità, si riesce, di conseguenza, a rilevare l'esistenza di informazioni steganografiche all'interno delle immagini.

Attacco statistico sul formato JPEG

Andreas Westfeld e Andreas Pfitzman notarono che, sistemi steganografici che cambiano i biti meno significativi, possono causare distorsioni che possono essere rilevate mediante la Steganalisi. In particolare, essi hanno notato che, per una data immagine, l'embedding dei dati, che presentano un'alta entropia (spesso dovuta alla cifratura cui è sottoposto il messaggio prima di essere sottoposto a Steganografia), cambiano l'istogramma delle frequenze dei colori in maniera prevedibile. Supponiamo di trovarci in un caso semplice, in cui il

Processo di Embedding sostituisce il bit meno significativo del colore dell'immagine con il bit del messaggio da incorporare. I colori sono indirizzati nella tabella dei colori attraverso i loro indici i ; definiamo le rispettive frequenze di occorrenza prima e dopo l'embedding come n_i e n_i^* . Dati dei bit di un messaggio da incorporare, distribuiti in modo uniforme, se $n_{2i} > n_{2i+1}$, allora i pixel di colore 2_i sono sostituiti con il colore 2_{i+1} più spesso rispetto al fatto che i pixel con colore 2_{i+1} vengano sostituiti con il colore 2_i . Di conseguenza, è molto probabile che valga la seguente relazione:

$$|n_{2i} - n_{2i+1}| \geq |n_{2i}^* - n_{2i+1}^*| \quad (2.4)$$

In altre parole, l'embedding dei bit del messaggio, distribuiti in maniera uniforme, riduce la differenza tra colori adiacenti. Lo stesso accade per il formato JPEG. Invece di misurare le frequenze di occorrenza dei colori, bisogna osservare che ci sono delle differenze tra le frequenze dei coefficiente DCT.

La seguente figura, presa da [20], mostra l'istogramma delle frequenze prima e dopo l'inserimento di un messaggio segreto in un'immagine JPEG.

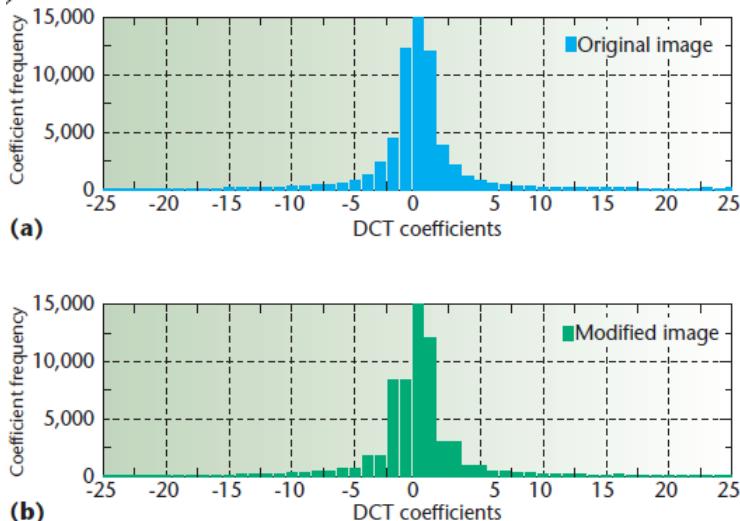


Figura 2.4: Differenza degli istogrammi prima e dopo l'embedding.

E' possibile notare una riduzione nelle differenze di frequenza tra il coefficiente DCT -1 ed il suo coefficiente adiacente -2. La stessa cosa si verifica tra i coefficienti 2 e 3.

Westfeld e Pfitzman hanno usato il cosiddetto χ^2 -test per determinare se la distribuzione y_i osservata in un'immagine corrisponde ad una distribuzione y_i^* , che presenta una distorsione causata dall'incorporamento di dati steganografici. Anche se non conosciamo la "cover image", sappiamo che la somma dei coefficienti DCT adiacenti rimane invariata e questo ci consente di calcolare la distribuzione y_i^* derivante dallo "stego image".

Sia n_i il numero totale di occorrenze del coefficiente i , calcoliamo la media aritmetica

$$y_i^* = \frac{n_{2i} + (n_{2i} + 1)}{2} \quad (2.5)$$

per determinare la distribuzione prevista e confrontarla con la distribuzione osservata $y_i = n_{2i}$.

Il valore χ^2 per la differenza tra le distribuzioni è data dalla seguente relazione:

$$\chi^2 = \sum_{i=1}^{\nu+1} \frac{y_i - y_i^*}{y_i^*} \quad (2.6)$$

dove ν è il grado di libertà, ovvero il numero delle diverse categorie presenti nell'istogramma. Sarebbe opportuno sommare i valori adiacenti della distribuzione prevista e della distribuzione osservata per garantire che ciascuna categoria abbia abbastanza peso. Combinare due categorie adiacenti, riduce il grado di libertà di 1.

La probabilità p che due distribuzioni siano uguali, è data dal complemento ad uno della distribuzione cumulativa:

$$p = 1 - \int_0^{\chi^2} \frac{t^{\frac{\nu-2}{2}} e^{-\frac{t}{2}}}{2^{\frac{\nu}{2}} \Gamma^{\frac{\nu}{2}}} dx \quad (2.7)$$

dove Γ è la funzione Gamma di Eulero.

La probabilità che sia presente un messaggio nascosto si determina calcolando p per un campione dell'insieme di tutti i coefficienti presenti all'inizio dell'immagine, poi per ogni misura successiva, il campione è man mano incrementato. In ogni caso sembra che il test non dia risultati positivi se i dati sono distribuiti in maniera del tutto casuale, come accade per l'algoritmo F5, di cui parleremo a breve.

2.1.5 Alcune applicazioni della steganografia sulle immagini

Di sotto, è riportata un'applicazione, che ci sarà utile nel corso della nostra trattazione. In particolare, esso è un'applicazione pratica della tecnica steganografica "**Algoritmi e Trasformazioni**", che operano, come abbiamo visto in precedenza, nel dominio della frequenza e che sfruttano i coefficienti della DCT per l'embedding dell'informazione steganografica.

L'algoritmo F5

L'algoritmo F5 è stato sviluppato nel 2001 ed effettua, essenzialmente, l'embedding di informazioni steganografiche all'interno di immagini JPEG. Abbiamo poc'anzi rivelato che l'algoritmo F5 non è rilevabile attraverso il test χ^2 . Infatti, esso incorpora i bit dei messaggi nei coefficienti DCT scelti casualmente ed utilizza il metodo Matrix Encoding, il cui obiettivo è quello di minimizzare il numero di modifiche necessarie per effettuare l'embedding di un messaggio segreto di una certa lunghezza. Secondo la descrizione dell'algoritmo F5 (versione 11), esso accetta 6 *input*:

1. Fattore di qualità Q dell'immagine stego;
2. File di input (TIFF, BMP, JPEG o GIF);
3. Nome del file di output;
4. File contenente il messaggio segreto;
5. Password utente da usare come seme per il generatore di numeri pseudo-random (PRNG);
6. Commento da inserire nell'intestazione.

L'algoritmo F5 effettua, in linea di massima, i seguenti passaggi:

- i coefficienti DCT vengono permutati attraverso il generatore di numeri pesudocasuali, utilizzando come seme per la generazione la chiave fornita dall'utente in input;
- successivamente, i coefficienti vengono raggruppati in gruppi di n elementi, saltando i coefficienti DC e quelli uguali a zero;
- il messaggio segreto è diviso in blocchi di k bit;
- per ogni blocco m del messaggio, si ottiene una parola a di n bit da concatenare con il bit meno significativo del valore assoluto del coefficiente DCT preso in esame;
- se il blocco del messaggio m e la decodifica $f(a)$ coincidono, allora il blocco del messaggio può essere incorporato senza eventuali modifiche; altrimenti, si usa la formula $s = m \oplus f(a)$ per determinare quale coefficiente debba essere modificato. La modifica consiste nel decrementare di 1 il valore assoluto del coefficiente: se quest'ultimo diventa zero, si verifica il cosiddetto "shrinkage", ovvero "restringimento". In questo caso F5 non lo riconosce come valore steganografico, pertanto, lo scarta e passa al successivo coefficiente diverso da zero tale da non produrre, a sua volta, uno zero.
- il processo si ripete fino a quando tutto il messaggio segreto non viene incorporato.

```

Input: message, shared secret, cover image
Output: stego image
initialize PRNG with shared secret
permute 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
         $G \leftarrow \{n \text{ non-zero AC coefficients}\}$ 
         $s \leftarrow k\text{-bit hash of LSB in } G$ 
         $s \leftarrow s \oplus k\text{-bit message block}$ 
        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 2.5: Processo di embedding eseguito dall'algoritmo F5 preso da [20].

Come abbiamo già anticipato in precedenza, l'algoritmo F5 non può essere rilevato né attraverso un attacco visivo, né attraverso un attacco statistico. Per ulteriori dettagli sull'algoritmo F5, fare riferimento a [20], [25].

Attacco all'algoritmo F5

Tuttavia, Jessica Fridrich ed i suoi colleghi hanno presentato un possibile attacco pensato per battere l'algoritmo F5. La loro idea consiste nello stimare l'istogramma della "cover image" e quello dello "stego image" e confrontare statisticamente i due istogrammi. Come risultato, hanno visto che è possibile ottenere un valore β (detto "*tasso di modifica*"),

che indica se F5 ha effettivamente modificato un'immagine. Tale metodo rileva come il processo di embedding di F5 cambi i coefficienti AC della "cover image". Sia

$$h_{uv}(d) := |\{F(u, v) | : d = |F(u, v)|, \quad u + v \neq 0\}| \quad (2.8)$$

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

$$H_{uv}(d) < (1 - \beta)h_{uv}(d) + \beta h_{uv}(d + 1) \quad \text{per } d > 0 \quad (2.9)$$

$$H_{uv}(0) > h_{uv}(0) + \beta h_{uv}(1) \quad \text{per } d = 0 \quad (2.10)$$

Fridrich ed il suo gruppo hanno usato questa stima per calcolare il *tasso di modifica* β previsto. Hanno trovato la miglior corrispondenza quando hanno usato $d = 0$ e $d = 1$, in quanto i valori di questi coefficienti cambiano molto durante il processo di embedding. Questo porta, pertanto, alla seguente approssimazione:

$$\beta_{uv} = \frac{h_{uv}(1)[H_{uv}(0) - h_{uv}(0)] + [H_{uv}(1) - h_{uv}(1)][H_{uv}(2) - h_{uv}(1)]}{h_{uv}^2(1) + [h_{uv}(2) - h_{uv}(1)]^2} \quad (2.11)$$

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

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

Per ulteriori dettagli riguardanti l'attacco dell'algoritmo F5 fare riferimento a [15].

Di sotto è riportato un esempio di applicazione pratico dell'algoritmo F5 eseguito su un'immagine JPEG. In particolare, la prima immagine rappresenta la "cover image", cioè l'immagine senza contenuto steganografico; la seconda è lo "stego image", quindi, l'immagine con contenuto steganografico; la terza immagine presenta le differenze tra la "cover image" e lo "stego image". Il messaggio segreto inserito, è la stringa "*ciao*", come password è stata scelta la stringa "*p*", come fattore di qualità $Q = 75$.⁵

⁵Se la "cover image" e lo "stego image" fossero stati uguali, avremmo avuto un'immagine differenza completamente nera. Il fatto che non lo sia, vuol dire che nella "cover image" è stato inserito del contenuto steganografico.



Figura 2.6: Cover Image



Figura 2.7: Stego Image

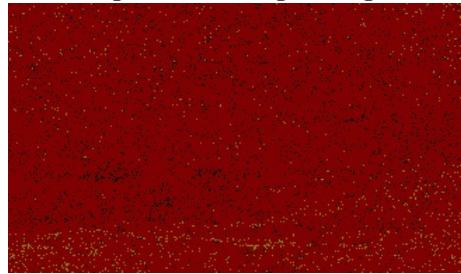


Figura 2.8: Differenza tra la "cover image" e lo "stego image"

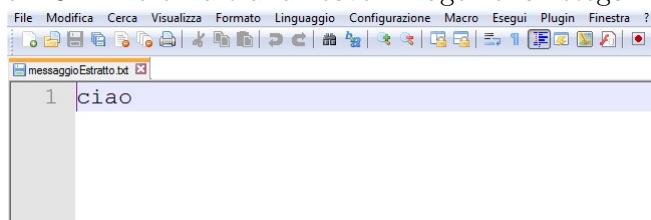


Figura 2.9: Messaggio estratto

2.2 La Steganografia sul Testo

⁶A differenza della Steganografia sulle immagini, quella sul testo è il tipo di Steganografia più difficile da utilizzare, in quanto un file di testo, a differenza delle immagini, è privo di informazioni "ridondanti" e, per questo motivo, non suscita grande interesse (perché, come vedremo in seguito, è facilmente attaccabile). La Steganografia sul testo utilizza, quindi, come "cover medium" e "stego medium" un file testuale, quindi del testo, per incorporare le informazioni steganografiche; pertanto, parleremo rispettivamente di "*cover text*" e di "*stego text*". Nel seguente paragrafo, il nostro obiettivo principale è quello di fornire una panoramica generale sulle tecniche steganografiche testuali attualmente esistenti e dei corrispettivi "attacchi".

⁶Il seguente paragrafo fa riferimento a [6] e [11].

La Steganografia sul Testo può essere essenzialmente classificata in 3 principali categorie:

1. Steganografia Linguistica

A sua volta, si suddivide in

- (a) *Metodo Sintattico*
- (b) *Metodo Semantico*

2. Generazione Casuale e Statistica

3. Steganografia Basata sul Formato

A sua volta, si suddivide in

- (a) *Codifica con spostamento di parole*
- (b) *Codifica con spostamento di righe*
- (c) *Codifica con spazio bianco*
- (d) *Codifica delle caratteristiche*

2.2.1 Steganografia linguistica

La Steganografia Linguistica considera le proprietà linguistiche del testo generato e modificato ed, in molti casi, utilizza la struttura linguistica come spazio in cui incorporare le informazioni steganografiche. Tuttavia, questo metodo presenta alcuni inconvenienti:

- se si fa uso di una "piccola" grammatica, ciò porterà senz'altro a molte ripetizioni testuali;
- inoltre, sebbene il testo sia sintatticamente corretto, potrebbe mancare di una struttura semantica, quindi, si potrebbero avere, per esempio, delle frasi senza senso oppure che non abbiano alcuna connessione tra loro.

Metodo Sintattico

Questa tecnica consiste nel modificare le caratteristiche sintattiche del testo come, ad esempio, l'utilizzo dei segni di punteggiatura per nascondere le informazioni steganografiche. Un esempio, di applicazione del metodo sintattico potrebbe essere il seguente:

"pane, burro, e latte" e "pane, burro e latte".

Entrambe le frasi sono corrette sintatticamente, sebbene la prima contenga una virgola in più. L'utilizzo di una delle due forme può rappresentare dei dati binari: ad esempio, ogni volta che verrà utilizzata la prima forma (che presenta una virgola prima della lettera "e"), questa verrà interpretata con il bit "1", mentre nel caso in cui venga utilizzata la seconda forma, questa verrà interpretata con il bit "0". Tuttavia, sebbene queste due forme siano corrette, potrebbe succedere, in alcuni casi, che il cambiamento della punteggiatura potrebbe avere un impatto considerevole sulla chiarezza o, persino, sul significato del testo e, inoltre, potrebbe essere facilmente rilevato da un eventuale lettore umano.

Metodo Semantico

Questa tecnica modifica la struttura semantica del testo per nascondere eventuali messaggi nascosti. Ad esempio, prendendo come riferimento la lingua inglese, potremmo utilizzare le parole "big" e "large", quindi dei *sinonimi*, per incorporare le informazioni steganografiche. In particolare, ogni qualvolta troveremo la parola "big", la interpreteremo come un "1", mentre ogni qualvolta troveremo la parola "large", la interpreteremo come uno "0". Tuttavia, bisogna prestare attenzione all'utilizzo dei sinonimi, in quanto in molti casi, potrebbero stravolgere radicalmente il significato stesso di una o più frasi.

2.2.2 Generazione casuale e statistica

Al fine di evitare un confronto con la "cover text" nota, si fa ricorso spesso alla generazione di "cover text", adatte a contenere le informazioni steganografiche da incorporare: la "cover text", quindi, viene generata a partire dal testo steganografico da incorporare. Il metodo consiste, in pratica, nell'incorporare le informazioni in una sequenza casuale di caratteri oppure nell'utilizzare le proprietà statistiche di una data lingua che fanno ricorso, ad esempio, alla lunghezza delle parole oppure alla frequenza delle lettere (come, ad esempio, le lettere più utilizzate nella lingua italiana).

2.2.3 Steganografia basata sul formato

I metodi steganografici basati sul formato del testo sfruttano la formattazione fisica del testo per nascondere le informazioni steganografiche, come, ad esempio, l'utilizzo di eventuali spazi bianchi tra le parole, tra le righe oppure alla fine di ciascuna riga, l'utilizzo di caratteri *non stampabili*, errori ortografici volutamente voluti nel testo, la modifica di alcuni caratteri, ecc... Questo metodo presenta, tuttavia, degli svantaggi:

- se il file di testo viene aperto con un elaboratore di testo, verranno rilevati eventuali errori ortografici così come la presenza di spazi bianchi extra;
- la modifica della dimensione dei caratteri, ad esempio, potrebbe insospettire eventuali lettori;
- se è disponibile il testo in chiaro originale, il confronto tra il testo originale ed il presunto testo, contenente le informazioni steganografiche, potrebbe portare ad evidenziare eventuali modifiche "visibili" del testo.

Codifica con spostamento di parole

Il messaggio segreto può essere nascosto spostando orizzontalmente le parole, ovvero verso destra o verso sinistra, per indicare rispettivamente i bit "0" oppure "1". Lo spostamento delle parole, in genere, può essere effettuato attraverso un metodo che è in grado di individuare se il blocco centrale è stato spostato verso destra oppure verso sinistra. Sebbene questo metodo ha poche possibilità di essere rilevato, se qualcuno conosce l'algoritmo ed, in particolare, la misura della distanza tra le parole utilizzate, si può facilmente risalire al messaggio nascosto nel testo. Inoltre, la riscrittura o l'utilizzo di un programma di riconoscimento dei caratteri potrebbe distruggere le informazioni steganografiche nascoste.

Codifica con spostamento di righe

Il messaggio può essere nascosto spostando le righe verticalmente, ovvero verso l'alto o verso il basso, per indicare rispettivamente i bit "0" e "1". Per determinare se una linea è stata spostata verso l'alto o verso il basso, viene preso come punto di riferimento la distanza rispetto la riga centrale. Tuttavia, anche in questo caso, se il testo viene riscritto oppure viene utilizzato un programma di riconoscimento dei caratteri, le informazioni steganografiche verrebbero distrutte. Inoltre, per poter utilizzare questo metodo, vi è la necessità di avere a disposizione una grande quantità di testo.

Codifica con spazio bianco

Questa tecnica sfrutta l'utilizzo di spazi bianchi extra per nascondere le informazioni steganografiche e possiamo individuare 3 metodi principali:

1. *spazio bianco tra le frasi*: un singolo spazio verrà interpretato come "0", mentre l'utilizzo di due spazi verrà interpretato come "1";

2. *spazi bianchi di fine riga*: viene inserito un numero fisso di spazi bianchi alla fine di una o più righe: ad esempio, 2 spazi per codificare un bit per riga, 4 spazi per codificare 2 bit e così via...
3. *spazi bianchi tra le parole*: un singolo spazio tra le parole verrà interpretato come "0", mentre l'utilizzo di due spazi tra le parole verrà interpretato come "1".

Tuttavia, l'utilizzo dello spazio bianco extra non è del tutto trasparente: è facilmente rilevabile da eventuali lettori umani.

Codifica delle caratteristiche

Il messaggio segreto può essere nascosto modificando alcune caratteristiche dei caratteri presenti nel testo. Queste caratteristiche possono essere, ad esempio, le linee verticali delle lettere "b", "d", "h", "k", ecc... , oppure i punti delle lettere "i", "j" o, ancora, le linee orizzontali delle lettere "f" e "t". Il difetto principale di questo metodo consiste nel fatto che, se si fa uso di un programma di riconoscimento dei caratteri o si esegue la riscrittura del testo, il contenuto nascosto, anche in questo caso, verrebbe distrutto.

2.2.4 Applicazioni attualmente esistenti di steganografia sul testo

1. I file HTML e XML

I file HTML e XML possono anche essere usati per nascondere informazioni. In particolare, se ci sono molti *tag di apertura e di chiusura*, questo viene interpretato attraverso il bit "0", se, invece, viene utilizzato un *singolo tag di apertura e di chiusura*, questo viene interpretato attraverso il bit "1". Un altro modo di utilizzare i file HTML e XML, per incorporare le informazioni steganografiche, consiste nello sfruttare lo spazio in un tag: in particolare, se c'è una mancanza di spazio, questo viene interpretato come uno "0", altrimenti (cioè se c'è uno spazio all'interno di un tag) questo viene interpretato come un "1". Per ulteriori dettagli fare riferimento a [6].

2. Gli SMS

La Steganografia può sfruttare anche gli SMS per inserire dei messaggi nascosti, in particolare, facendo uso di *abbreviazioni*: possiamo utilizzare la parola per intero oppure una sua abbreviazione per nascondere informazioni steganografiche. Praticamente, per nascondere il bit "0", viene utilizzata la sua forma completa e, per nascondere il bit "1", la sua forma abbreviata. Per ulteriori dettagli fare riferimento a [6].

3. L'utilizzo di caratteri "simili" in un file MS Word

Si sfrutta la somiglianza di alcuni font per i caratteri come, ad esempio, i font *Century751BT*, *CenturyOldStyle* e *CenturyExpdBT*. Un esempio di font uguali è riportato nella seguente figura (presa da [24]).

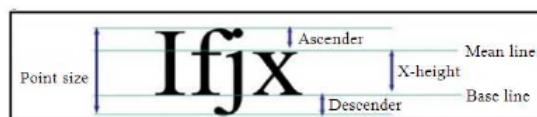


Figura 2.10: Esempio di font simili.

In genere, questa tecnica funziona soprattutto nel caso di un lettore umano. Infatti, se si utilizzano dei programmi di riconoscimento dei caratteri oppure MS Word stesso, è possibile rilevare l'utilizzo di font diversi per i caratteri. Per ulteriori dettagli fare riferimento a [24].

4. L'utilizzo di un colore diverso per i caratteri

In questo approccio si sfrutta il *il colore del font* per nascondere dei messaggi segreti. In particolare, si fa uso di due alfabeti simili, ma che sono colorati con colori RGB successivi o simili. Pertanto, almeno visivamente risulta impossibile stabilire le differenze tra i toni di colore utilizzati, ma un'analisi più attenta, però, potrebbe rilevarle. Per ulteriori dettagli fare riferimento a [12].



Figura 2.11: Esempio di colori RGB successivi o simili.

5. L'utilizzo di caratteri "non stampabili"

Questo approccio fa uso di due caratteri non stampabili:

(a) Zero width nonJoiner (ZWNJ)

Il carattere *Zero width non-joiner* ("spazio a larghezza nulla"), che corrisponde al carattere "*U+200D*" nella tabella UNICODE, in genere, sta ad indicare l'assenza di spazio tra le parole. In questo contesto, però, marca la fine del messaggio segreto all'interno del testo.

(b) Zero width Joiner (ZWJ) ("spazio a larghezza non nulla"), che corrisponde al carattere "*U+200C*" nella tabella UNICODE, in genere, indica la presenza di spazio tra le parole. In questo contesto, però, marca le posizioni usate per incorporare le informazioni steganografiche.

Esempio di caratteri non stampabili — ZWJ

EsempioDiCaratteriNonStampabili — ZWNJ

Figura 2.12: Esempio di utilizzo dei caratteri ZWNJ e ZWJ.

Il vantaggio di utilizzare caratteri di questo tipo (non stampabili) è quello di non andare a modificare le proprietà del file, come ad esempio il formato oppure il contenuto; di conseguenza, la forma del testo rimane invariata e non solleva, dunque, sospetti, migliorandone la robustezza. Prima di poter utilizzare questa tecnica, però, è necessario verificare la capacità del file contenitore, in quanto potrebbe succedere che la "cover text" non abbia spazio sufficiente per contenere il messaggio. Per effettuare una verifica di questo tipo, bisogna semplicemente verificare che la dimensione del "cover text" sia maggiore rispetto alla lunghezza del messaggio segreto da incorporare. Per ulteriori dettagli fare riferimento a [22].

6. Proprietà di rotazione simmetrica degli alfabeti

Per nascondere un messaggio all'interno del testo, questa tecnica fa ricorso all'utilizzo della *Proprietà di Rotazione Simmetrica degli Alfabeti*. In pratica, si individua un *asse*

di simmetria (verticale, orizzontale, entrambi oppure nessuno dei due) e si effettua la *bisezione* sulle lettere: quindi, le lettere vengono divise a metà in base all'asse di simmetria scelto. E' possibile, pertanto, individuare 4 gruppi:

- (a) **Lettere che non hanno nessuna simmetria:** C, F, G, J, L, N, P, Q, R, Z. A queste lettere vengono assegnati i bit "00" per nascondere le informazioni steganografiche.
- (b) **Lettere che presentano una simmetria lungo l'asse orizzontale:** B, D, E, K, S. A queste lettere vengono assegnati i bit "01" per nascondere le informazioni steganografiche.
- (c) **Lettere che presentano una simmetria lungo l'asse verticale:** A, M, T, U, V, W, Y. A queste lettere vengono assegnati i bit "10" per nascondere le informazioni steganografiche.
- (d) **Lettere che presentano una simmetria su entrambi gli assi:** H, I, O, X. A queste lettere vengono assegnati i bit "11" per nascondere le informazioni steganografiche.

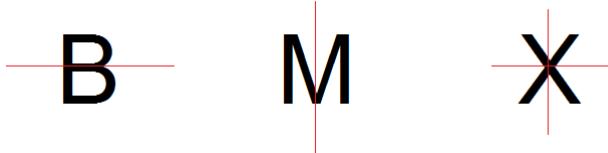


Figura 2.13: Esempio pratico di lettere cui è stata applicata la "*Proprietà di Rotazione Simmetrica*".

Per ulteriori dettagli fare riferimento a [9].

7. L'utilizzo dell'operatore "TJ" nel PDF

In genere, i file PDF presentano un vantaggio importante, ovvero quello di essere indipendenti dall'hardware, dal sistema operativo e dai software utilizzati per la loro visualizzazione. Nonostante, presentino queste caratteristiche, è possibile sfruttare il posizionamento del dispositivo di destinazione per nascondere delle informazioni steganografiche. La posizione del testo nei file PDF viene effettuata attraverso il "*Modello di Posizionamento Grafico Vettoriale 2D*", in particolare, sfruttando l'"operatore TJ", che è utilizzato per "disegnare", appunto, il testo in una pagina PDF. In particolare, è possibile utilizzare la stringa dell'operatore TJ per nascondere i dati. Per ulteriori dettagli fare riferimento a [23].

AWAY again	<small>[(AWAY again) TJ</small>
AWAY again	<small>[(A) 120 (W) 120 (A) 95 (Y again) TJ</small>

Figura 2.14: Operazione dell'operatore TJ.

8. Utilizzo di simboli "invisibili" in un documento MS Word

Questa tecnica permette di nascondere delle informazioni steganografiche utilizzando principalmente *4 simboli "invisibili"*. In genere, si utilizza una chiave per generare una *Tabella dei Simboli*, utilizzata per avere una corrispondenza tra i dati da nascondere ed i corrispondenti simboli.

I 4 simboli utilizzati in questo algoritmo sono i seguenti:

- (a) **Marcatore da sinistra a destra**, noto anche come *LRM* ("Left-to-right mark"), che corrisponde a *U+200E* nella tabella UNICODE.
- (b) **Marcatore da destra a sinistra**, noto come *RLM* ("Right-to-left mark"), che corrisponde a *U+200F* nella tabella UNICODE.
- (c) **Spazio a larghezza non nulla**, noto come *ZWNJ* ("zero width non-joiner"), che corrisponde a *U+200C* nella tabella UNICODE.
- (d) **Spazio a larghezza nulla**, noto come *ZWJ* ("zero width joinere", che corrisponde a *U+200D* nella tabella UNICODE.

A seconda se siano presenti tutti i simboli, solamente uno, due oppure tre, si ha una differente corrispondenza binaria utilizzata per nascondere le informazioni steganografiche. Di sotto, è riportata la tabella utilizzata per l'inserimento delle informazioni steganografiche.

Right Remark	Left Remark	ZWJ	ZWNJ	Hidden code
X	X	X	X	0000
X	X	X		0001
X		X		0101
X	X			0011
X				0111
X	X	X	X	1111
X	X		X	1101

Figura 2.15: Tabella dei Simboli utilizzata per l'embedding delle informazioni steganografiche in un documento MS Word presa da [5].

In particolare, se sono presenti tutti e 4 i simboli, allora la stringa binaria utilizzata per nascondere l'informazione steganografica sarà "0000", se sono presenti i simboli RLM, LRM e ZWJ, allora la stringa binaria utilizzata per nascondere l'informazione steganografica sarà "0001", e così via... Dal momento che la tabella dei simboli viene generata a partire da una chiave segreta, il destinatario è in grado di risalire alle informazioni nascoste se e solo se conosce tale chiave. Il vantaggio principale di questa tecnica risiede principalmente nel fatto che essa non va ad interferire sulle proprietà del file, come ad esempio la dimensione, il contenuto ed il formato ed, inoltre, può essere estesa a qualsiasi lingua. Per ulteriori dettagli fare riferimento a [5].

Capitolo

3

METODI E IMPLEMENTAZIONE

Questo capitolo può essere considerato il "*cuore*" della presente tesi, dal momento che viene illustrata la nuova tecnica di Steganografia sul testo.

Come è stato sottolineato nel **Capitolo 2**, la *Steganografia sul testo*, rispetto alla Steganografia sulle immagini, è la forma di steganografia più difficile da realizzare, dal momento che di per sé il testo non offre informazioni ridondanti; di conseguenza, è più facile rilevare delle anomalie al suo interno, causate dall'inserimento di informazioni steganografiche.

In genere, affinché possa esserci uno scambio di messaggi segreti tra un mittente ed un destinatario, è necessario scegliere con cura il "mezzo" in cui inserire l'informazione steganografica, che nel nostro caso specifico sarà un file PDF, detto "**cover PDF**". Alla fine del processo steganografico, che illustreremo a breve, si otterrà il cosiddetto "**stego PDF**", che dovrà contenere l'informazione steganografica ed, allo stesso tempo, dovrà essere il più possibile simile al "cover PDF", in modo tale da non destare sospetti agli occhi di un eventuale osservatore o ficcanaso.

Per prima cosa, forniremo una breve descrizione del formato PDF, dopodiché verrà illustrata questa nuova tecnica steganografica sul PDF.

3.1 Il Formato PDF ("Portable Document Format")

¹Il formato PDF ("Portable Document Format") è un formato di file basato su un linguaggio di descrizione di pagina (il *PostScript*) sviluppato da *Adobe System* nel 1993 per rappresentare documenti in maniera tale da risultare del tutto indipendenti dalle applicazioni software, dall'hardware e dal sistema operativo utilizzati per crearli o per visualizzarli. La sua versione attuale è la 1.7, come è riportato da [18].

A differenza del linguaggio PostScript, il PDF non è un linguaggio di programmazione e, di conseguenza, non contiene procedure, variabili o costrutti di controllo; esso definisce una propria serie di operatori di alto livello che sono sufficienti per descrivere la maggior parte delle pagine.

Un documento PDF può contenere una o più pagine, ciascuna delle quali può contenere al suo interno una combinazione qualsiasi di testo, grafici ed immagini in un formato del tutto indipendente dal dispositivo e dalla risoluzione. Inoltre, può anche contenere dei collegamenti ipertestuali, segnalibri, la versione specifica del PDF utilizzata ed informazioni sulla posizione di importanti strutture nel file.

¹Il seguente paragrafo è tratto da [14].

3.1.1 Gli oggetti del formato PDF

Il PDF supporta *7 tipi di oggetti di base*: i valori booleani, i numeri, le stringhe, i nomi, gli array, i dizionari e gli stream. Inoltre, fornisce anche un *oggetto NULL*. Gli oggetti possono essere etichettati in maniera tale da poter essere indicati da altri oggetti. Un oggetto etichettato è chiamato *oggetto indiretto* (*o "indirect object"*).

NOTA: Il PDF è *case-sensitive*, ovvero fa distinzione tra le lettere maiuscole e le lettere minuscole.

1. Boolean

Gli oggetti di tipo *boolean* sono rappresentati dalla parola chiave *true* e *false*.

2. Numeri

Il PDF fornisce due tipi di numeri: *interi* e *reali*. I numeri interi possono essere specificati da costanti con o senza segno. I numeri reali possono essere solo in formato esadecimale.

3. Stringhe

Una stringa è una sequenza di caratteri delimitati da parentesi. Se una stringa è troppo lunga per essere posizionata su una singola riga, può essere suddivisa su più righe usando il carattere backslash ("\\") alla fine di una riga per indicare che la stringa continua sulla riga successiva. In questo caso i caratteri *backslash* e di *fine linea* non sono considerati parte della stringa. All'interno di una stringa il carattere *backslash* può essere usato per specificare parentesi sbilanciate, caratteri ASCII non stampabili ed il carattere backslash stesso. Questo meccanismo è lo stesso usato nelle stringhe del linguaggio PostScript. Come nel linguaggio PostScript, anche le stringhe possono essere rappresentate in forma esadecimale. Una stringa esadecimale è costituita da una sequenza di caratteri esadecimali (le cifre 0-9 e le lettere A-F oppure a-f) racchiuse tra parentesi angolari (<e>). Ogni coppia di cifre esadecimali definisce un carattere della stringa. Se manca la cifra finale, si assume che essa sia zero. I caratteri degli spazi bianchi (spazio, tabulazione, ritorno a capo, avanzamento di riga e avanzamento modulo) vengono ignorati.

Per esempio: <901fa3> è una stringa di 3 caratteri composta da caratteri, i cui codici esadecimali sono 90, 1f, a3. Al contrario, <901fa> è una stringa di 3 caratteri contenente caratteri, i cui codici esadecimali sono 90, 1f, a0.

Le stringhe possono essere usate per vari scopi e possono essere formattate in diversi modi.

4. Nomi

Un nome, come una stringa, è una sequenza di caratteri. Deve iniziare con uno *slash* seguito da una sequenza di caratteri ASCII nell'intervallo !(<21>) a ~(<7E) tranne %, (,), <, >, [,], , , \, #. In un nome può essere incluso qualsiasi carattere (tranne *null*), scrivendo il suo codice esadecimale a due caratteri, preceduto da #.

NOTA: Il numero massimo di caratteri in un nome è 127. Questo limite si riferisce alla rappresentazione interna del nome.

5. Array

Un array è una sequenza di oggetti PDF e può contenere una combinazione di tipi di oggetti. Esso viene scritto come una sequenza di una parentesi quadra sinistra "[", seguita da una sequenza di oggetti, seguiti da una parentesi quadra destra "]". Un esempio di array è il seguente: [0 (Higgs) false 3.14 3 549 \SomeName].

6. Dizionario

Un dizionario è una tabella associativa che contiene coppie di oggetti. Il primo elemento di ogni coppia è chiamato *chiave* ed il secondo elemento è chiamato *valore*. A differenza dei dizionari del linguaggio PostScript, una chiave deve essere un *nome*. Un valore può essere un qualsiasi tipo di oggetto, incluso un dizionario. Un dizionario viene generalmente utilizzato per raccogliere e legare insieme gli attributi di un oggetto complesso con ciascuna coppia *chiave-valore*, che specifica il nome ed il valore di un attributo. Un dizionario è rappresentato da 2 parentesi angolari sinistre («), seguite da una sequenza di coppie *chiave-valore*, seguite da 2 parentesi angolari destre (»).

Esempio di dizionario:

```
« /Type, /Example, /key2, 12, /key3, (a string) »
```

Gli oggetti dizionario sono *gli elementi principali* di un documento PDF. Molti componenti di un PDF, come le pagine ed i caratteri, sono rappresentate tramite dizionari. Per convenzione la chiave *Type* di un dizionario specifica il tipo di oggetto descritto dal dizionario ed il suo valore è sempre un *nome*. In alcuni casi, la chiave *Subtype* è usata per descrivere una specializzazione di un tipo particolare ed il suo valore è sempre un *nome*. Ad esempio per un font, *Type* è *Font* ed esistono numerosi *Subtype*, tra cui *Type1*, *MMType1*, *Type3*, *TrueType* ed altri.

7. Stream

Uno stream, come una stringa, è una sequenza di caratteri. Tuttavia, un'applicazione può leggere una piccola porzione di stream alla volta, mentre una stringa deve essere letta nella sua interezza. Per questo motivo, gli oggetti con una quantità potenzialmente elevata di dati, come immagini e descrizioni di pagina, sono rappresentati come stream. Uno stream è costituito da un *dizionario* che descrive una sequenza di caratteri, dalla parola chiave *stream*, seguita da zero o più righe di caratteri, e dalla parola chiave *endstream*. Tutti gli *stream* devono essere *oggetti indiretti*. Lo stream *dizionario* deve essere un *oggetto diretto*. La parola chiave *stream*, che segue lo stream *dizionario*, deve essere seguita da *un ritorno a capo ed un avanzamento di riga* oppure *solo da un avanzamento di riga*. Una sequenza di caratteri, che compongono uno stream, si può trovare tra le parole chiave *stream* ed *endstream* oppure può essere contenuta in un *file esterno*. Se i dati si trovano in un file esterno, lo stream *dizionario* specifica il *file*. Quando i dati di uno stream sono *esterni*, i caratteri tra *stream* ed *endstream* vengono *ignorati*.

3.1.2 L'oggetto NULL

L'oggetto NULL è rappresentato dalla parola chiave *null*.

NOTA: il valore di una chiave nel *dizionario* può essere specificata come *null*. Un modo più semplice per esprimere ciò, è quello di omettere la chiave del dizionario.

3.1.3 L'oggetto indiretto e l'oggetto diretto

Un *oggetto diretto* è un *valore booleano*, un *numero*, una *stringa*, un *nome*, un *array*, un *dizionario*, uno *stream* o *null*.

Un *oggetto indiretto* è un oggetto che è stato etichettato in modo tale da poter essere referenziato da altri oggetti.

Qualsiasi tipo di oggetto può essere etichettato come oggetto indiretto. Gli oggetti indiretti sono molto utili: ad esempio, se la lunghezza di uno stream non è nota prima della

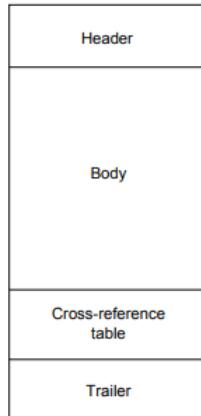
sua scrittura, il valore della chiave *Length* dello stream può essere specificato come oggetto indiretto archiviato nel file dopo lo stream. Un oggetto indiretto è costituito da un identificatore *object*, da *un oggetto diretto* e la parola chiave *endobj*. L'identificatore *object* è formato da un *numero intero*, un *numero di generazione intero* e la parola chiave *obj*. La combinazione del numero dell'oggetto e del numero di generazione serve come identificativo univoco per un oggetto indiretto. Per tutta la sua esistenza un oggetto indiretto conserva il numero dell'oggetto ed il numero di generazione, che gli era stato inizialmente assegnato, anche se l'oggetto è stato modificato. Ogni oggetto indiretto ha un numero di oggetto univoco spesso, ma non necessariamente, numerati in sequenza, a partire da 1. Fino a quando un oggetto nel file non viene eliminato, tutti i numeri di generazione sono zero.

3.1.4 Riferimenti agli oggetti

Qualsiasi oggetto, usato come elemento di un array o come valore in un dizionario, può essere specificato da un oggetto diretto o da un riferimento indiretto. Un *riferimento indiretto* è un riferimento ad un oggetto indiretto ed è costituito dal *numero dell'oggetto indiretto*, dal *numero di generazione* e dalla parola chiave *R*. Un riferimento indiretto ad un oggetto non definito non è un errore: viene trattato come un *riferimento all'oggetto null*. Ad esempio, se un file PDF contiene il riferimento indiretto *(12 0 R)*, ma non contiene la definizione *(12 0 obj ... endobj)*, il riferimento indiretto è nullo.

3.1.5 Struttura di un file PDF

Un file PDF presenta essenzialmente la seguente struttura:



1. Header

L'*header* è la prima riga di un file PDF che specifica il *numero di versione* delle specifiche PDF a cui un file è conforme.

Ad esempio, se un file PDF ha versione 1.3, esso è conforme ad una *versione precedente* (quindi, esso sarà conforme anche ad una versione 1.0, 1.1, 1.2).

2. Body

Il body (o *corpo*) di un file PDF contiene una *sequenza di oggetti indiretti* che rappresentano il *contenuto* del documento.

3. Cross-reference table

La cross-reference table (o *tabella dei riferimenti incrociati*) contiene le informazioni che consentono l'*accesso casuale* agli oggetti indiretti nel file; pertanto, non è necessario leggere l'intero file per individuare un oggetto particolare. Per ogni oggetto

indiretto nel file la tabella contiene una voce di una riga che descrive la *posizione* dell'oggetto nel file. Un file PDF contiene una tabella di riferimenti composta da una o più *sezioni*. Se non sono stati fatti degli aggiornamenti al file, la tabella dei riferimenti contiene una singola sezione. Viene aggiunta una sezione ogni volta che vengono effettuati degli aggiornamenti al file. La sezione di riferimento è l'unica parte del PDF con un *formato fisso*; ciò ne permette l'accesso casuale alle voci nella tabella dei riferimenti. La sezione inizia con una riga contenente la parola chiave *xref*.

4. Trailer

Il *trailer* consente ad un'applicazione che legge un file PDF di trovare rapidamente la tabella dei riferimenti ed alcuni oggetti speciali. L'*ultima riga* di un file PDF contiene solo l'*indicatore di fine file* %%EOF. Le due righe precedenti contengono le parole chiave *startxref* ed il *byte dell'offset* dall'inizio del file all'inizio della parola chiave *xref* nell'ultima sezione della tabella dei riferimenti nel file. Il *dizionario del trailer* precede questa linea. Esso è costituito dalla parola chiave *trailer* seguita da un insieme di coppie *chiave-valore* racchiuse tra parentesi quadre. Se il file non ha subito aggiornamenti, non troviamo la parola chiave *Prev* nel dizionario del trailer.

3.1.6 Struttura di un documento PDF

Il PDF fornisce una rappresentazione elettronica di un documento: una serie di pagine contenenti testo, grafici ed immagini, insieme ad altre informazioni, come ad esempio annotazioni di testo, collegamenti ipertestuali e segnalibri. Ai fini della nostra trattazione, ci soffermeremo sul *contenuto* della sezione *Body*.

Il documento PDF può essere descritto come una *gerarchia di oggetti* contenuti nella sezione *body*. La maggior parte degli oggetti di questa gerarchia sono *dizionari*. Le *relazioni parent* ("genitore"), *child* ("figlio") e *sibling* ("fratello/sorella") sono rappresentate da coppie *chiave-valore*, i cui valori sono riferimenti indiretti a oggetti parent, child o sibling. Ad esempio, l'oggetto *CATALOG*, che è la *radice* ("root") della gerarchia, contiene una chiave *Pages*, il cui valore è un riferimento indiretto all'oggetto che è la radice dell'albero *Pages*. Il trailer del file PDF "TRAILER" specifica la posizione dell'oggetto *CATALOG* come valore della chiave *ROOT* del trailer. Inoltre, il trailer specifica la posizione del dizionario delle informazioni del documento, una struttura che contiene informazioni generali sul documento, come il valore della chiave *Info* del trailer.

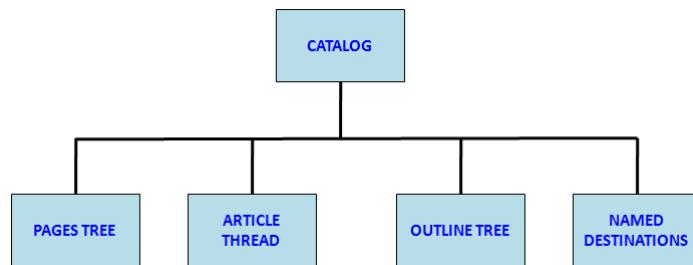


Figura 3.1: Struttura di un documento PDF.

Catalog (o catalogo)

Il catalogo è un *dizionario* che è il *nodo radice* ("root node") del documento. Contiene un riferimento all'*albero delle pagine* ("tree of pages") contenuto nel documento, un riferimento

all'*albero degli oggetti*, che rappresentano la struttura del documento, un riferimento ai *thread degli articoli* del documento e l'*elenco delle destinazioni con nome*.

Pages tree (o albero delle pagine)

Le pagine di un documento sono accessibili attraverso un *albero di nodi*, noto come *albero delle pagine ("Pages tree")*. Questo albero definisce l'*ordinamento delle pagine* nel documento. Alcune applicazioni di visualizzazione, per ottimizzare le prestazioni, costruiscono degli *alberi bilanciati*. La *struttura ad albero* consente alle applicazioni di aprire rapidamente un documento contenente migliaia di pagine. Le applicazioni devono accettare qualsiasi tipo di struttura ad albero. La struttura più semplice consiste in un *singolo nodo Pages* che fa riferimento direttamente a tutti gli oggetti della pagina. La radice e tutti i nodi interni dell'albero *Pages* sono *dizionari*. Un oggetto *Pages* può contenere *chiavi aggiuntive* che forniscono dei valori per gli oggetti *Page*, che sono i suoi discendenti. Si dice che tali valori sono "*ereditati*".

Page objects (o pagina degli oggetti)

Un oggetto *Page* è un *dizionario*, le cui chiavi descrivono una singola pagina contenente testo, grafici ed immagini. Un oggetto *Page* è una *foglia* dell'albero *Pages*.

Le destinazioni

Una voce *annotazione* o *OUTLINE* può specificare una *destinazione*, che consiste in una pagina, una posizione della finestra di visualizzazione su quella pagina ed un fattore zoom da usare quando si visualizza quella pagina. Una destinazione può essere rappresentata esplicitamente come un *array* o implicitamente attraverso un *nome*.

Name tree (o albero dei nomi)

Il catalogo di un documento può contenere una chiave *Names*, il cui valore è un *dizionario*. Ogni valore di questo dizionario è un *albero dei nomi*, che è un albero, simile all'albero *Pages*, in cui i nodi foglia contengono coppie di stringhe (i "nomi") ed oggetti (i "valori"). L'albero dei nomi ha lo stesso scopo di un dizionario, ovvero quello di mappare le chiavi ai valori, ma lo fa in modo diverso: le chiavi in un albero dei nomi sono stringhe, non oggetti *Name* del PDF.

Article thread (o thread di articoli)

Un thread di articoli identifica gli elementi correlati in un documento, consentendo ad un utente di seguire un flusso di informazioni che può comprendere più colonne o pagine. Un documento PDF può includere uno o più thread di articoli. Ogni thread ha un *titolo* ed un *elenco di elementi Thread*. Se un documento contiene dei thread, questi vengono archiviati in un array come il valore *Thread* nell'oggetto *Catalog*. Ogni thread è un *dizionario*.

3.1.7 Resources dictionaries (o dizionari delle risorse)

Le operazioni di marcatura per disegnare una pagina sono memorizzate in uno stream, che è il valore della chiave *Contents* nel dizionario dell'oggetto *Page*. Le operazioni di marcatura usano vari tipi di oggetti di base, come i numeri e le stringhe; questi sono rappresentati come oggetti diretti nello stream dei *Contents*. Altri oggetti, come i font, che sono rappresentati da stream o dizionari, possono essere anche necessari per le operazioni di marcatura, ma nessun oggetto indiretto di alcun tipo, inclusi gli stream, può apparire nello stream *Contents*. Ogni content di marcatura include una *lista di risorse* che viene

archiviata come un *dizionario*, che è il valore della chiave *Resources* del content e svolge due funzioni: elenca le risorse denominate nello stream *Contents* e stabilisce una corrispondenza tra i nomi e gli oggetti usati dalle operazioni di marcatura. Il PDF fornisce diversi tipi di *Resources*. Questi includono:

- **ProcSet**: è un *array* di nomi predefiniti;
- **Font**: è un *dizionario* in cui ogni chiave è una risorsa *name* ed ogni valore è un *Font*;
- **ColorSpace**: è un *dizionario* in cui ogni chiave è una risorsa *name* ed ogni valore è il nome di un colore, che dipende dal dispositivo, o un *array* che descrive un colore;
- **XObjects**: è un *dizionario* in cui ogni chiave è una risorsa *name* ed ogni valore è un *XObject*;
- **Extended graphics state**: è un *dizionario* in cui ogni chiave è una risorsa *name* ed ogni valore è un dizionario *Extended graphics state*;
- **Pattern**: è un *dizionario* in cui ogni chiave è una risorsa *name* ed ogni valore è un *Pattern*;
- **Property List**: è un *dizionario* in cui ogni chiave è una risorsa *name* ed ogni valore è una *Property List*;
- **Shading dictionary**: è un *dizionario* in cui ogni chiave è una risorsa *name* ed ogni valore è un dizionario *Shading*.

3.1.8 XObject

Gli XObject sono denominati *Resources* ("Risorse"). Il PDF attualmente supporta 3 tipi di XObject: le *immagini*, le *form* ed i *frammenti del linguaggio PostScript pass-through*.

Le immagini

Un'immagine è un oggetto *XObject*, il cui *Subtype* ("sottotipo") è *Images*, che consente ad un processo di marcatura di specificare un'immagine campionata ("sample image") o la maschera di un'immagine ("mask image"). Il PDF supporta la *mask image*, le *immagini a scala di grigi* a 1 bit, 2 bit, 4 bit e a 8 bit, ed *immagini a colori* con 1, 2, 4 oppure 8 bit per componente. Le immagini a colori possono avere un componente *Color* (valori di componenti indicizzati o tinte di separazione), 3 componenti *Color* (RGB, CalRGB o Lab), 4 componenti *Color* (CMYK) o un numero arbitrario (DeviceN).

Un XObject *Image* è specificato da un oggetto *Stream*. Il dizionario *Stream* deve includere la chiavi standard richieste per tutti gli stream, nonché quelle aggiuntive descritte di seguito.

Una *MaskImage* è un valore che, se è pari ad 1, indica dove il colore deve essere applicato, se è pari a 0, indica che non è necessario applicare alcun colore in quella posizione; pertanto, il colore in quella determinata posizione è lasciato così com'è. Una *mask image* può essere usata anche per indicare dove un'altra immagine dovrebbe essere "dipinta" e può specificare quali colori devono essere "dipinti" e quali no in un'immagine.

Alcuni *attributi* dell' *XObject Image*:

- **Type:** è un *nome* ed è un tipo di oggetto *XObject*;
- **Subtype:** è un *nome*, è un sottotipo di *XObject* ed è di tipo *Image*;
- **Name:** è un *nome* della risorsa che deve corrispondere al nome usato nel dizionario *XObject* all'interno del dizionario delle risorse delle pagine;
- **Width:** è un *numero intero* e rappresenta la larghezza dell'immagine sorgente campionata;
- **Height:** è un *numero intero* e rappresenta l'altezza dell'immagine sorgente campionata.

3.2 Steganografia su PDF

Supponiamo di voler nascondere un messaggio segreto all'interno di un documento PDF. Per il nostro processo steganografico avremo bisogno essenzialmente di due processi principali: il **processo di embedding** ed il **processo di estrazione**.

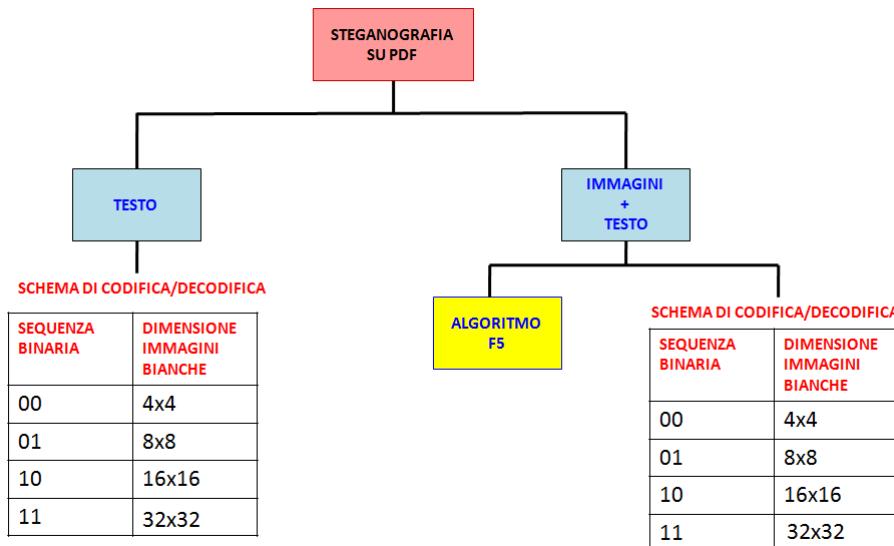


Figura 3.2: Steganografia su PDF.

Tuttavia, bisogna tener ben presente che un documento PDF può contenere sia del testo, sia una combinazione di immagini e di testo. A seconda che il documento contenga la prima o la seconda opzione, il processo di embedding ed il processo di estrazione effettueranno degli step differenti.

3.2.1 Processo di embedding

Il *processo di embedding* prende in *input* un *cover PDF*, una *password* ed il *messaggio segreto* e ci fornisce come *output* lo *stego PDF*, che contiene l'informazione steganografica da inviare al destinatario.

Esaminiamo separatamente il caso in cui il documento PDF contenga solamente testo ed il caso in cui il documento PDF contenga una combinazione di testo e di immagini.

- Il PDF contiene solo contenuto testuale

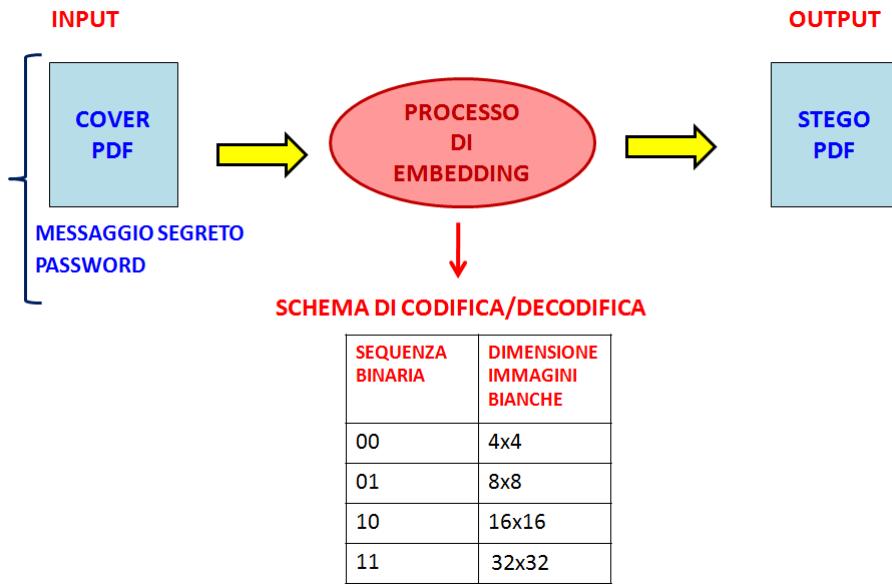


Figura 3.3: Schema generico del processo di embedding quando il PDF contiene del contenuto testuale.

Se il documento PDF ha solamente del contenuto testuale, si procede all'inserimento del messaggio segreto seguendo il seguente **schema di codifica**:

1. si converte il messaggio segreto in messaggio binario;
2. la sequenza binaria ottenuta si suddivide in sottosequenze di 2 bit;

NOTA: per la conversione in binario si è considerata la tabella ASCII; pertanto, ciascun carattere è una sequenza di 8 bit.

A seconda della sottosequenza, presa in considerazione, avremo 4 scenari possibili:

- (a) se la sottosequenza è **00**, la sostituiremo con un'**immagine bianca** di dimensione **4x4**;
 - (b) se la sottosequenza è **01**, la sostituiremo con un'**immagine bianca** di dimensione **8x8**;
 - (c) se la sottosequenza è **10**, la sostituiremo con un'**immagine bianca** di dimensione **16x16**;
 - (d) se la sottosequenza è **11**, la sostituiremo con un'**immagine bianca** di dimensione **32x32**.
3. infine, al posto della sottosequenza sostituiremo un'immagine bianca, come riportato nel passaggio precedente.

- Il PDF contiene una combinazione di immagini e di testo

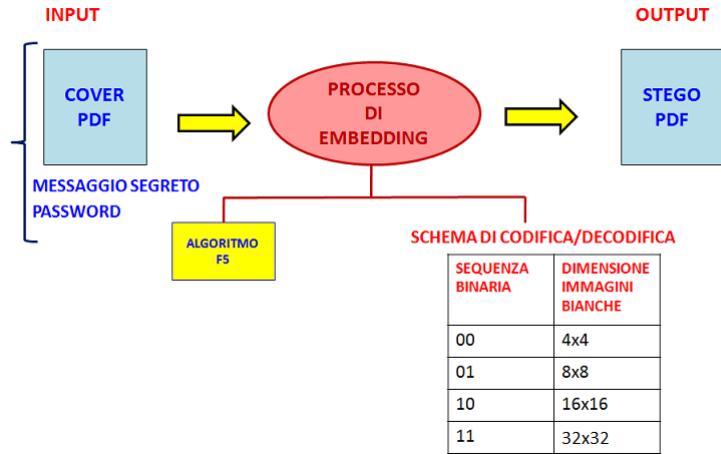


Figura 3.4: Schema generico del processo di embedding quando il PDF contiene una combinazione di immagini e di testo.

Se, invece, il documento PDF contiene una combinazione di immagini e di testo, si procede come segue:

1. il messaggio segreto viene inserito nelle immagini contenute nel PDF, utilizzando l'algoritmo F5, descritto nel **Capitolo 2**;
2. infine, si effettua l'inserimento del messaggio segreto, seguendo gli stessi step visti quando il PDF contiene solo del contenuto testuale.

3.2.2 Processo di estrazione

Il *processo di estrazione*, invece, prende come *input* uno *stego PDF* ed una *password* e ci fornisce come *output* il *messaggio segreto* contenuto al suo interno.

Come per il *processo di embedding*, possiamo avere due scenari diversi:

- Il PDF contiene solo contenuto testuale

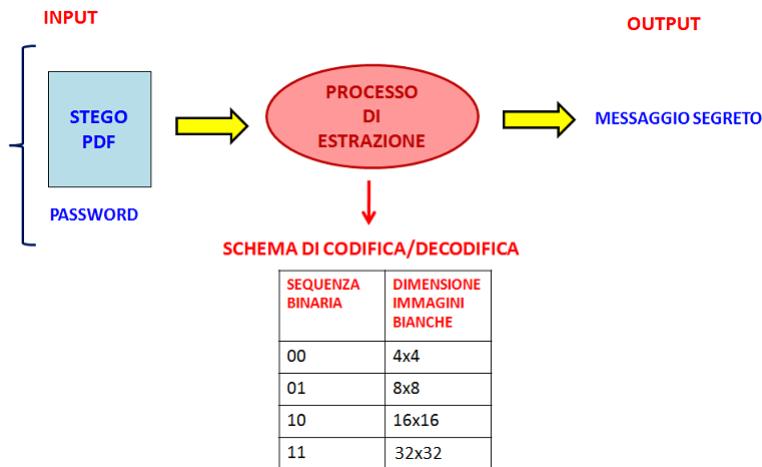


Figura 3.5: Schema del processo di estrazione quando il PDF contiene del contenuto testuale.

Se il PDF contiene solamente del contenuto testuale, verifichiamo per prima cosa che il documento PDF in questione contenga delle immagini bianche di dimensione 4x4, 8x8, 16x16, 32x32.

- se il PDF non contiene immagini di questo tipo, vuol dire che il documento PDF non contiene alcuna informazione steganografica;
- se il PDF contiene immagini di questo tipo, si procede come segue:
 1. si estraggono le immagini bianche;
 2. a seconda della loro dimensione, si effettua la seguente sostituzione:
 - * se l'immagine bianca ha dimensione **4x4**, si sostituisce con una sequenza di 2 bit pari a **00**;
 - * se l'immagine bianca ha dimensione **8x8**, si sostituisce con una sequenza di 2 bit pari a **01**;
 - * se l'immagine bianca ha dimensione **16x16**, si sostituisce con una sequenza di 2 bit pari a **10**;
 - * se l'immagine bianca ha dimensione **32x32**, si sostituisce con una sequenza di 2 bit pari a **11**.
 3. una volta attribuita a ciascuna immagine la corrispondente sequenza binaria di 2 bit, si ottiene il messaggio binario contenuto nel documento PDF;
 4. si converte il messaggio binario secondo la tabella ASCII, considerando il fatto che ciascun carattere è una sequenza di 8 bit; di conseguenza, il messaggio binario estratto sarà suddiviso in sottosequenze di 8 bit, ciascuna delle quali corrisponderà ad un determinato carattere nella tabella ASCII.

- Il PDF contiene una combinazione di immagini e di testo

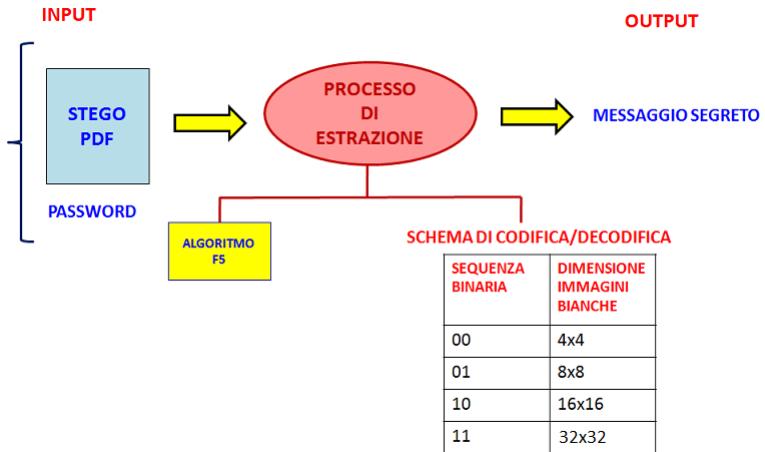


Figura 3.6: Schema del processo di estrazione quando il PDF contiene una combinazione di immagini e di testo.

Se il documento PDF contiene una combinazione di immagini e di testo, si procede in questo modo:

1. Si estraggono le immagini presenti nel documento PDF, tranne le immagini bianche di dimensioni 4x4, 8x8, 16x16, 32x32;
2. si procede con l'estrazione del messaggio segreto dalle immagini, contenute nel documento PDF, utilizzando l'**algoritmo F5**, descritto nel **Capitolo 2**;
3. infine, si estraе il messaggio segreto, eseguendo gli stessi step visti quando il PDF contiene solo del contenuto testuale.

3.3 Aspetti Implementativi

Per poter implementare un processo steganografico di questo tipo, è stata utilizzata la libreria **iText**² che fornisce una serie di metodi per la lettura, la scrittura e la modifica di file PDF. Il **codice sorgente Java**, relativo all'applicazione sviluppata per un processo steganografico di questo tipo, è disponibile su <https://github.com/rocchinaRomano?tab=repositories>, nella repository “steganografiaSuPDF”, all'interno della quale è possibile trovare il file “**README.txt**”, nel quale vi sono tutte le informazioni necessarie per il download e l'utilizzo dell'applicazione.

²*iText* è una libreria, disponibile sia per Java, sia per .NET. La sua versione attualmente disponibile è la versione 7 ed è distribuita sotto licenza **Affero General Public License (AGPL)** [26].

All'avvio dell'applicazione "**Steganografia su PDF**" verrà mostrata una schermata, che permetterà all'utente di scegliere se effettuare l'**inserimento di un messaggio segreto** in un documento PDF oppure l'**estrazione di un messaggio segreto** a partire da un documento PDF.

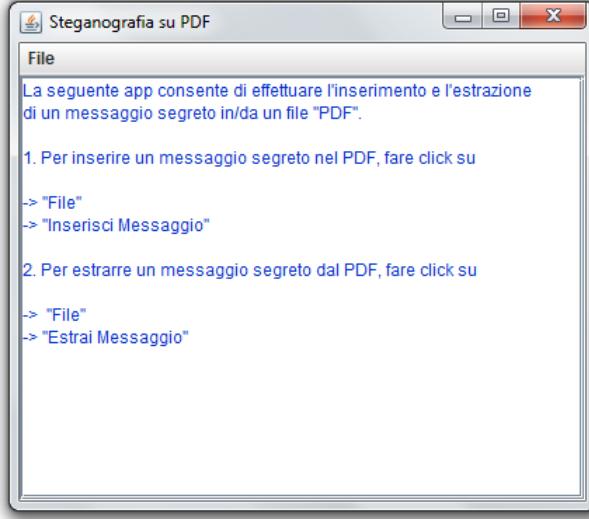


Figura 3.7: Schermata principale dell'applicazione "Steganografia su PDF."

A seconda che l'utente voglia effettuare l'una oppure l'altra operazione, il sistema mostrerà due schermate diverse, come mostriremo a breve.

3.3.1 Processo di embedding

Se l'utente decide di **inserire un messaggio segreto** in un documento PDF, il sistema mostrerà la seguente schermata:

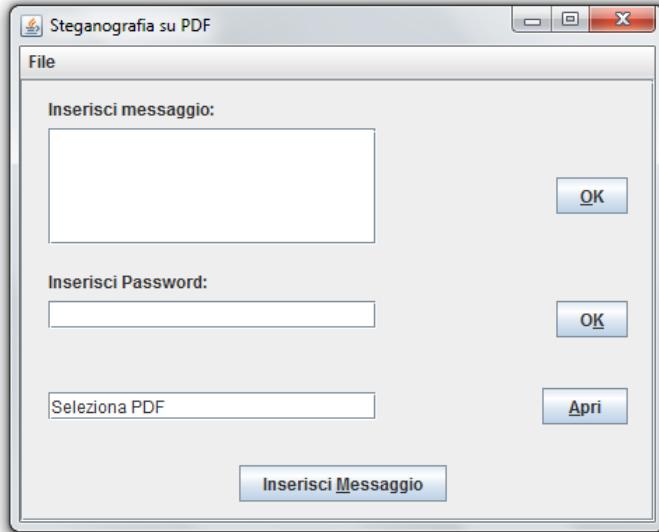


Figura 3.8: Schermata di Embedding dell'applicazione "Steganografia su PDF."

Supponiamo che l'utente voglia inserire in un determinato documento PDF il messaggio "*ciao*" e la password "*pass*", dopo avere selezionato un *cover PDF*. Come abbiamo già

illustrato in precedenza, a seconda che il documento abbia solo contenuto testuale oppure abbia sia del testo che immagini, effettuerà l'embedding del messaggio segreto in maniera differente. Analizziamo separatamente i **2 casi**.

Embedding in un documento contenente solo del testo

Se il documento PDF contiene solamente del testo, il processo di embedding effettuerà essenzialmente i seguenti step:

- il messaggio segreto è convertito in messaggio binario;
- il messaggio binario, ottenuto allo step precedente, è suddiviso in sottosequenze di 2 bit, dove ciascuna sottosequenza di 2 bit è inserita in una lista di stringhe;
- dopo aver inizializzato la lista delle sottosequenze, si procede con l'inserimento delle immagini bianche, seguendo lo **schema di codifica**, visto in precedenza. In particolare, per ogni elemento della lista delle sottosequenze, si inserisce l'immagine bianca corrispondente in ogni pagina del PDF. Questo processo continua fino a quando non abbiamo esaminato tutti gli elementi presenti nella lista delle sottosequenze.

NOTA: è bene evidenziare che per l'inserimento delle immagini bianche all'interno del PDF, è necessario fornire delle **coordinate**. La prima coordinata, scelta a questo proposito, è la coordinata ($x1 = 1$; $y1 = 773$). Nel momento in cui la coordinata $x1$ supera il valore 550, si procede con l'inserimento delle immagini a partire dalla riga successiva. Questo passaggio è fondamentale per evitare che le immagini inserite trasbordino dai margini della pagina del documento PDF.

- Come output del seguente *processo di embedding*, otterremo lo **stego PDF**.

NOTA: la classe **PdfStamper** di *iText* permette di inserire le immagini sotto il contenuto di una pagina del documento PDF, attraverso il metodo **getUnderContent(int numeroPagina)**, in modo tale che non vada ad oscurare il suo contenuto originale. Per evidenziare questa cosa, forniremo sia un esempio di *stego PDF*, facendo uso di immagini colorate, anziché bianche, e successivamente forniremo un esempio dello stesso *stego PDF*, al cui interno sono, invece, state inserite delle immagini bianche. Tuttavia, **PdfStamper** possiede anche un metodo che permette di inserire il contenuto sovrapponendolo a quello già esistente attraverso il metodo **getOverContent(int numeroPagina)**.



**Regolamento Didattico del
Corso di Laurea Magistrale Interstruttura in
Ingegneria Informatica e delle Tecnologie dell'Informazione
(Classe LM-32 Ingegneria Informatica)
Anno Accademico 2019-2020**

*Approvato dal:
Consiglio dei Corsi di Studio del 19.2.2019
Consiglio della Scuola di Ingegneria del 27.2.2019*

Art. 1 - Finalità

- Il presente Regolamento disciplina il Corso di Laurea Magistrale Interstruttura in Ingegneria Informatica e delle Tecnologie dell'Informazione (CdLM-III) (Classe LM-32, Ingegneria Informatica), corso di studi interstruttura attivato, ai sensi del D.M. n. 270/04, dalla Scuola di Ingegneria e dal Dipartimento di Matematica, Informatica ed Economia (di seguito denominate Strutture di riferimento) dell'Università degli Studi della Basilicata (USB). Sede amministrativa del corso di studi è la Scuola di Ingegneria.
- Detto Regolamento, deliberato dalla Scuola di Ingegneria in conformità con l'ordinamento didattico nel rispetto della libertà d'insegnamento, nonché dei diritti e doveri dei docenti e degli studenti, specifica gli aspetti collegati alla didattica del CdLM-III, ai sensi dell'art. 12 del D.M. n. 270/04. In particolare: l'elenco degli insegnamenti (con l'indicazione dei settori scientifico-disciplinari e dell'articolazione) e delle altre attività formative, gli obiettivi formativi specifici, i crediti e le eventuali propedeuticità di insegnamenti e attività formative, la tipologia delle forme didattiche, le modalità di accesso, i requisiti di ammissione.
- Gli ulteriori aspetti di carattere organizzativo collegati al corso di studi sono stabiliti in un apposito *Regolamento di Funzionamento del Consiglio dei Corsi di Studi Interstruttura in Scienze e Tecnologie Informatiche e Ingegneria Informatica e delle Tecnologie dell'Informazione* (di seguito denominato Consiglio dei Corsi di Studi Interstruttura o CCStI).
- Per quanto concerne ogni altro aspetto di carattere organizzativo, il CdLM-III si attiene a quanto espressamente disciplinato dai Regolamenti delle strutture di riferimento e di Ateneo.

Art. 2 - Organi Didattici di Riferimento

- Il Corso di Laurea Magistrale è retto dal Consiglio dei Corsi di Studio Interstruttura in Scienze e Tecnologie Informatiche e Ingegneria Informatica e delle Tecnologie dell'Informazione.
- La composizione e la funzione del suddetto Consiglio di Corso di Studi sono stabilite nei Regolamenti di Funzionamento delle Strutture di riferimento e nel *Regolamento di Funzionamento del Consiglio di Corso di Studi Interstruttura*.

Art. 3 - Curricula e figure professionali

- L'articolazione del CdLM-III è così definita:

Attività formativa	S.S.D.	CFU
Caratterizzante	ING-INF/04, ING-INF/05	45
Affini	ING-INF/01, ING-INF/02, ING-INF/03, ING-IND/31, INF/01, FIS/06	48
A Scuola		9
Ubazioni Attivita'		6
Prova Finale		12

1

**Regolamento Didattico del
Corso di Laurea Magistrale Interstruttura in
Ingegneria Informatica e delle Tecnologie dell'Informazione
(Classe LM-32 Ingegneria Informatica)
Anno Accademico 2019-2020**

*Approvato dal:
Consiglio dei Corsi di Studio del 19.2.2019
Consiglio della Scuola di Ingegneria del 27.2.2019*

Art. 1 - Finalità

- Il presente Regolamento disciplina il Corso di Laurea Magistrale Interstruttura in Ingegneria Informatica e delle Tecnologie dell'Informazione (CdLM-III) (Classe LM-32, Ingegneria Informatica), corso di studi interstruttura attivato, ai sensi del D.M. n. 270/04, dalla Scuola di Ingegneria e dal Dipartimento di Matematica, Informatica ed Economia (di seguito denominate Strutture di riferimento) dell'Università degli Studi della Basilicata (USB). Sede amministrativa del corso di studi è la Scuola di Ingegneria.
- Detto Regolamento, deliberato dalla Scuola di Ingegneria in conformità con l'ordinamento didattico nel rispetto della libertà d'insegnamento, nonché dei diritti e doveri dei docenti e degli studenti, specifica gli aspetti collegati alla didattica del CdLM-III, ai sensi dell'art. 12 del D.M. n. 270/04. In particolare: l'elenco degli insegnamenti (con l'indicazione dei settori scientifico-disciplinari e dell'articolazione) e delle altre attività formative, gli obiettivi formativi specifici, i crediti e le eventuali propedeuticità di insegnamenti e attività formative, la tipologia delle forme didattiche, le modalità di accesso, i requisiti di ammissione.
- Gli ulteriori aspetti di carattere organizzativo collegati al corso di studi sono stabiliti in un apposito *Regolamento di Funzionamento del Consiglio dei Corsi di Studi Interstruttura in Scienze e Tecnologie Informatiche e Ingegneria Informatica e delle Tecnologie dell'Informazione* (di seguito denominato Consiglio dei Corsi di Studi Interstruttura o CCStI).
- Per quanto concerne ogni altro aspetto di carattere organizzativo, il CdLM-III si attiene a quanto espressamente disciplinato dai Regolamenti delle strutture di riferimento e di Ateneo.

Art. 2 - Organi Didattici di Riferimento

- Il Corso di Laurea Magistrale è retto dal Consiglio dei Corsi di Studio Interstruttura in Scienze e Tecnologie Informatiche e Ingegneria Informatica e delle Tecnologie dell'Informazione.
- La composizione e la funzione del suddetto Consiglio di Corso di Studi sono stabilite nei Regolamenti di Funzionamento delle Strutture di riferimento e nel *Regolamento di Funzionamento del Consiglio di Corso di Studi Interstruttura*.

Art. 3 - Curricula e figure professionali

- L'articolazione del CdLM-III è così definita:

Attività formativa	S.S.D.	CFU
Caratterizzante	ING-INF/04, ING-INF/05	45
Affini	ING-INF/01, ING-INF/02, ING-INF/03, ING-IND/31, INF/01, FIS/06	48
A Scuola		9
Ubazioni Attivita'		6
Prova Finale		12

1

Figura 3.9: Lo "stego PDF" con immagini colorate e bianche ottenuto in seguito al processo di Embedding.

Di seguito riporteremo alcuni pezzi di **codice sorgente Java** utilizzati per il *processo di Embedding*.

```

1   private void convertiMessaggio(String messaggio) {
2       this.messBinario = "";
3       for(int i = 0; i < this.messaggio.length(); i++) {
4           String s = Integer.toBinaryString(this.messaggio.charAt(i));
5           while(s.length() < 8){
6               s = "0" + s;
7           }
8           this.messBinario = this.messBinario + s;
9       }
10      }
11
12
13     private void inizializzaListaSottoseq() {
14         String mess = this.messBinario;
15         int i = 0;
16         int j = 2;
17         while(mess.length() != 0){
18             String sottoseq = mess.substring(i, j);
19             this.listaSottosequenze.add(sottoseq);
20             String copy = mess;
21             mess = copy.substring(j, copy.length());
22         }
23     }
24
25     private void aggiungiSottosequenza(float x1, float y1, Image image,
26     String stegoPdf) {
27         try {
28             PdfReader reader = new PdfReader(this.coverAppoggio);
29             PdfStamper stamper = new PdfStamper(reader,
30                     new FileOutputStream(stegoPdf));
31             for(int i = 1; i <= reader.getNumberOfPages(); i++ ){
32                 image.setAbsolutePosition(x1, y1);
33                 stamper.getUnderContent(i).addImage(image);
34             }
35             stamper.close();
36             reader.close();
37         } catch (IOException | DocumentException ex) {
38             String errore = "ATTENZIONE!\n"
39             + "Errore_durante_il_salvataggio_del_file!";
40             this.steganografia.getSp().finestraErrore(errore);
41         }
42     }

```

```

1   private void inserisciMessaggio(){
2       this.numImmagini = 1;
3       try {
4           coverAppoggio = this.cover;
5           float x1 = 1;
6           float y1 = 773;
7           for(int i = 0; i < this.listaSottosequenze.size(); i++){
8               String sottoSeq = (String)this.listaSottosequenze.get(i);
9               String pdfStego = "steganografiasupdf/embedding/temp/stego";
10              if(sottoSeq.equalsIgnoreCase("00")){
11                  if(i == this.listaSottosequenze.size()-1){
12                      pdfStego = stego;
13                  } else{
14                      pdfStego = pdfStego + i + ".pdf";
15                  }
16                  Image image = Image.getInstance(immagine1);
17                  aggiungiSottosequenza(x1, y1, image, pdfStego);
18                  x1 = x1 + (float)0.5 + image.getWidth();
19                  this.coverAppoggio = pdfStego;
20              } else if(sottoSeq.equalsIgnoreCase("01")){
21                  if(i == this.listaSottosequenze.size()-1){
22                      pdfStego = stego;
23                  } else{
24                      pdfStego = pdfStego + i + ".pdf";
25                  }
26                  Image image = Image.getInstance(immagine2);
27                  aggiungiSottosequenza(x1, y1, image, pdfStego);
28                  x1 = x1 + (float)0.5 + image.getWidth();
29                  this.coverAppoggio = pdfStego;
30              } else if(sottoSeq.equalsIgnoreCase("10")){
31                  if(i == this.listaSottosequenze.size()-1){
32                      pdfStego = stego;
33                  } else{
34                      pdfStego = pdfStego + i + ".pdf";
35                  }
36                  Image image = Image.getInstance(immagine3);
37                  aggiungiSottosequenza(x1, y1, image, pdfStego);
38                  x1 = x1 + (float)0.5 + image.getWidth();
39                  this.coverAppoggio = pdfStego;
40              } else if(sottoSeq.equalsIgnoreCase("11")){
41                  if(i == this.listaSottosequenze.size()-1){
42                      pdfStego = stego;
43                  } else{
44                      pdfStego = pdfStego + i + ".pdf";
45                  }
46                  Image image = Image.getInstance(immagine4);
47                  aggiungiSottosequenza(x1, y1, image, pdfStego);
48                  x1 = x1 + (float)0.5 + image.getWidth();
49                  this.coverAppoggio = pdfStego;
50              }
51              if(x1 > 550){
52                  x1 = 1;
53                  y1 = y1 - 32 - 1;
54              }
55          }
56      } catch (IOException | BadElementException ex) {
57          String errore = "ATTENZIONE!!!\n"
58          + ("Errore_durante_il_processo_di_EMBEDDING!");
59          this.steganografia.getSp().finestraErrore(errore);
60      }
61  }

```

Embedding in un documento contenente sia immagini sia testo

In questo caso, il sistema effettuerà due **passaggi principali**:

1. effettuerà l'embedding attraverso l'**algoritmo F5** sulle immagini già presenti all'interno del documento PDF;
2. effettuerà l'embedding del messaggio segreto su tutte le pagine del PDF attraverso l'utilizzo delle immagini bianche, nello stesso modo in cui avviene quando il documento PDF contiene solamente del testo.

Verrà analizzato solamente il primo passaggio, in quanto il secondo è già stato analizzato precedentemente.

- Se il documento PDF contiene al suo interno delle immagini, per prima cosa è necessario estrarre tali immagini;
- per ogni immagine estratta, è necessario fare l'embedding del messaggio utilizzando l'algoritmo F5;
- infine, si sostituiscono le immagini originali del documento PDF con le *immagini stego*, derivanti dal processo di embedding eseguito con F5.
- Una volta effettuate le sostituzioni, si eseguirà l'inserimento del messaggio segreto attraverso l'utilizzo delle *immagini bianche*, nello stesso modo in cui avviene quando il documento PDF contiene solamente del testo;
- Come output del seguente *processo di embedding*, otterremo lo **stego PDF**.

Per la sostituzione delle immagini, è stato utilizzato il seguente codice sorgente Java:

```

1  private void sostituisci(int numPagina) {
2     try {
3         String appoggio;
4         if(pag == 1){
5             appoggio = stegoF5;
6         }else{
7             appoggio = pathTemp + "out_" + numPdfF5 + ".pdf";
8         }
9         PdfReader reader = new PdfReader(pdfAppoggioF5);
10        PdfStamper stamper = new PdfStamper(reader, new FileOutputStream(appoggio));
11        PdfWriter writer = stamper.getWriter();
12        PdfDictionary pg = reader.getPageN(pag);
13        PdfDictionary res = (PdfDictionary)PdfReader.getPdfObject(
14            pg.get(PdfName.RESOURCES));
15        PdfDictionary xobj = (PdfDictionary)PdfReader.getPdfObject(
16            res.get(PdfName.XOBJECT));
17        if(xobj != null){
18            int keys = xobj.getKeys().size();
19            Set<PdfName> setKey = xobj.getKeys();
20            int i = 0;
21            String[] resImage = new String[keys];
22            for(PdfName name : setKey){
23                resImage[i] = name.toString();
24                i++;
25            }
26            for(int j = 0; j <= resImage.length-1; j++){
27                String s = resImage[j];
28                for(PdfName name : setKey){
29                    String sPdf = name.toString();
30                    if(s.equals(sPdf)){
31                        PdfObject obj = xobj.get(name);
32                        if(obj.isIndirect()){
33                            PdfDictionary tg = (PdfDictionary)PdfReader.getPdfObject(obj);
34                            PdfName type = (PdfName)PdfReader.getPdfObject(
35                                tg.get(PdfName.SUBTYPE));
36                            if(PdfName.IMAGE.equals(type)){
37                                PdfReader.killIndirect(obj);
38                                String path = cercaImmagine(listaPathFileStego);
39                                Image img = Image.getInstance(path);
40                                Image imageMask = img.getImageMask();
41                                if(imageMask != null){
42                                    writer.addDirectImageSimple(imageMask);
43                                }
44                                writer.addDirectImageSimple(img,
45                                    (PRIIndirectReference)obj);
46                                eliminaImmagine(path);
47                                break;
48                            }
49                        }
50                    }
51                }
52            }
53            stamper.close();
54            reader.close();
55            pdfAppoggioF5 = appoggio;
56            numPdfF5++;
57        }catch (IOException | DocumentException ex) {
58            String errore = "ATTENZIONE!\n"
59            + "Errore_durante_il_caricamento_del_file!";
60            this.steganografia.getSp().finestraError5(errore);
61        }
62    }

```

NOTA: L'esempio riportato di sotto mostra uno *stego PDF* ottenuto applicando il processo di embedding appena descritto. In particolare, otterremo un risultato del tutto simile a quello che si ha quando il documento PDF contiene solamente del testo.

 UNIVERSITÀ DEGLI STUDI DELLA BASILICATA	 UNIVERSITÀ DEGLI STUDI DELLA BASILICATA												
<p>INFORMATIVA AGLI STUDENTI SUL TRATTAMENTO DEI DATI PERSONALI</p>													
<p>Ai sensi degli artt. 13 e 14 del Regolamento generale sulla protezione dei dati "Regolamento (UE) 2016/679 del Parlamento europeo e del Consiglio del 27 aprile 2016" (GDPR), l'Università degli Studi della Basilicata fornisce agli studenti già iscritti a qualunque tipologia di corso di studio e a quanti intendono immatricolarsi/iscriversi ai medesimi corsi di studio le seguenti informazioni relativamente all'utilizzo dei dati personali che li riguardano:</p>													
<table border="1"> <thead> <tr> <th style="text-align: center;">Titolare del trattamento</th> </tr> </thead> <tbody> <tr> <td>Titolare del trattamento è l'Università degli Studi della Basilicata, nella persona del Rettore pro tempore, con sede a Potenza (85100) - Via Nazario Sauro 85; e-mail: protocollo@pec.unibas.it.</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th style="text-align: center;">Responsabile della protezione dei dati</th> </tr> </thead> <tbody> <tr> <td>Dati di contatto del Responsabile della protezione dei dati: rpd@unibas.it.</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th style="text-align: center;">Finalità e base giuridica del trattamento</th> </tr> </thead> <tbody> <tr> <td>I dati personali, forniti dagli studenti all'atto dell'immatricolazione/iscrizione o durante lo svolgimento della carriera, sono trattati dall'Università degli Studi della Basilicata, nel rispetto dei principi di cui all'art. 5 del GDPR, ai soli fini istituzionali. In particolare, i dati saranno trattati per le seguenti finalità: a) immatricolazione/iscrizione ai corsi di studio e frequenza, sia in presenza sia in modalità e-learning; b) gestione della carriera universitaria; c) gestione dei servizi curricolari ed extracurricolari; d) mobilità nazionale e internazionale; e) determinazione del contributo onnicomprensivo annuale e applicazione delle agevolazioni in materia di tasse; f) conseguimento del titolo di studio; g) utilizzo dei servizi telematici e di posta elettronica; h) accesso ai laboratori e ad altre strutture protette; i) invio comunicazioni inerenti la carriera universitaria; k) applicazione delle misure di sicurezza degli ambienti di lavoro secondo le disposizioni del D.Lgs. 81/2008; l) procedimenti di natura disciplinare a carico di studenti; m) valutazioni per la valutazione della didattica e per la customer satisfaction; n) archiviazione e conservazione dati inerenti la carriera universitaria (studi svolti, incarichi ricoperti, titoli di studio conseguiti); o) elezioni rappresentanze studentesche ed eventuale svolgimento di cariche elettive negli</td> </tr> </tbody> </table>	Titolare del trattamento	Titolare del trattamento è l'Università degli Studi della Basilicata, nella persona del Rettore pro tempore, con sede a Potenza (85100) - Via Nazario Sauro 85; e-mail: protocollo@pec.unibas.it .	Responsabile della protezione dei dati	Dati di contatto del Responsabile della protezione dei dati: rpd@unibas.it .	Finalità e base giuridica del trattamento	I dati personali, forniti dagli studenti all'atto dell'immatricolazione/iscrizione o durante lo svolgimento della carriera, sono trattati dall'Università degli Studi della Basilicata, nel rispetto dei principi di cui all'art. 5 del GDPR, ai soli fini istituzionali. In particolare, i dati saranno trattati per le seguenti finalità: a) immatricolazione/iscrizione ai corsi di studio e frequenza, sia in presenza sia in modalità e-learning; b) gestione della carriera universitaria; c) gestione dei servizi curricolari ed extracurricolari; d) mobilità nazionale e internazionale; e) determinazione del contributo onnicomprensivo annuale e applicazione delle agevolazioni in materia di tasse; f) conseguimento del titolo di studio; g) utilizzo dei servizi telematici e di posta elettronica; h) accesso ai laboratori e ad altre strutture protette; i) invio comunicazioni inerenti la carriera universitaria; k) applicazione delle misure di sicurezza degli ambienti di lavoro secondo le disposizioni del D.Lgs. 81/2008; l) procedimenti di natura disciplinare a carico di studenti; m) valutazioni per la valutazione della didattica e per la customer satisfaction; n) archiviazione e conservazione dati inerenti la carriera universitaria (studi svolti, incarichi ricoperti, titoli di studio conseguiti); o) elezioni rappresentanze studentesche ed eventuale svolgimento di cariche elettive negli	<table border="1"> <thead> <tr> <th style="text-align: center;">Titolare del trattamento</th> </tr> </thead> <tbody> <tr> <td>Titolare del trattamento è l'Università degli Studi della Basilicata, nella persona del Rettore pro tempore, con sede a Potenza (85100) - Via Nazario Sauro 85; e-mail: protocollo@pec.unibas.it.</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th style="text-align: center;">Responsabile della protezione dei dati</th> </tr> </thead> <tbody> <tr> <td>Dati di contatto del Responsabile della protezione dei dati: rp@unibas.it.</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th style="text-align: center;">Finalità e base giuridica del trattamento</th> </tr> </thead> <tbody> <tr> <td>I dati personali, forniti dagli studenti all'atto dell'immatricolazione/iscrizione o durante lo svolgimento della carriera, sono trattati dall'Università degli Studi della Basilicata, nel rispetto dei principi di cui all'art. 5 del GDPR, ai soli fini istituzionali. In particolare, i dati saranno trattati per le seguenti finalità: a) immatricolazione/iscrizione ai corsi di studio e frequenza, sia in presenza sia in modalità e-learning; b) gestione della carriera universitaria; c) gestione dei servizi curricolari ed extracurricolari; d) mobilità nazionale e internazionale; e) determinazione del contributo onnicomprensivo annuale e applicazione delle agevolazioni in materia di tasse; f) conseguimento del titolo di studio; g) utilizzo dei servizi telematici e di posta elettronica; h) accesso ai laboratori e ad altre strutture protette; i) invio comunicazioni inerenti la carriera universitaria; k) applicazione delle misure di sicurezza degli ambienti di lavoro secondo le disposizioni del D.Lgs. 81/2008; l) procedimenti di natura disciplinare a carico di studenti; m) valutazioni per la valutazione della didattica e per la customer satisfaction; n) archiviazione e conservazione dati inerenti la carriera universitaria (studi svolti, incarichi ricoperti, titoli di studio conseguiti); o) elezioni rappresentanze studentesche ed eventuale svolgimento di cariche elettive negli</td> </tr> </tbody> </table>	Titolare del trattamento	Titolare del trattamento è l'Università degli Studi della Basilicata, nella persona del Rettore pro tempore, con sede a Potenza (85100) - Via Nazario Sauro 85; e-mail: protocollo@pec.unibas.it .	Responsabile della protezione dei dati	Dati di contatto del Responsabile della protezione dei dati: rp@unibas.it .	Finalità e base giuridica del trattamento	I dati personali, forniti dagli studenti all'atto dell'immatricolazione/iscrizione o durante lo svolgimento della carriera, sono trattati dall'Università degli Studi della Basilicata, nel rispetto dei principi di cui all'art. 5 del GDPR, ai soli fini istituzionali. In particolare, i dati saranno trattati per le seguenti finalità: a) immatricolazione/iscrizione ai corsi di studio e frequenza, sia in presenza sia in modalità e-learning; b) gestione della carriera universitaria; c) gestione dei servizi curricolari ed extracurricolari; d) mobilità nazionale e internazionale; e) determinazione del contributo onnicomprensivo annuale e applicazione delle agevolazioni in materia di tasse; f) conseguimento del titolo di studio; g) utilizzo dei servizi telematici e di posta elettronica; h) accesso ai laboratori e ad altre strutture protette; i) invio comunicazioni inerenti la carriera universitaria; k) applicazione delle misure di sicurezza degli ambienti di lavoro secondo le disposizioni del D.Lgs. 81/2008; l) procedimenti di natura disciplinare a carico di studenti; m) valutazioni per la valutazione della didattica e per la customer satisfaction; n) archiviazione e conservazione dati inerenti la carriera universitaria (studi svolti, incarichi ricoperti, titoli di studio conseguiti); o) elezioni rappresentanze studentesche ed eventuale svolgimento di cariche elettive negli
Titolare del trattamento													
Titolare del trattamento è l'Università degli Studi della Basilicata, nella persona del Rettore pro tempore, con sede a Potenza (85100) - Via Nazario Sauro 85; e-mail: protocollo@pec.unibas.it .													
Responsabile della protezione dei dati													
Dati di contatto del Responsabile della protezione dei dati: rpd@unibas.it .													
Finalità e base giuridica del trattamento													
I dati personali, forniti dagli studenti all'atto dell'immatricolazione/iscrizione o durante lo svolgimento della carriera, sono trattati dall'Università degli Studi della Basilicata, nel rispetto dei principi di cui all'art. 5 del GDPR, ai soli fini istituzionali. In particolare, i dati saranno trattati per le seguenti finalità: a) immatricolazione/iscrizione ai corsi di studio e frequenza, sia in presenza sia in modalità e-learning; b) gestione della carriera universitaria; c) gestione dei servizi curricolari ed extracurricolari; d) mobilità nazionale e internazionale; e) determinazione del contributo onnicomprensivo annuale e applicazione delle agevolazioni in materia di tasse; f) conseguimento del titolo di studio; g) utilizzo dei servizi telematici e di posta elettronica; h) accesso ai laboratori e ad altre strutture protette; i) invio comunicazioni inerenti la carriera universitaria; k) applicazione delle misure di sicurezza degli ambienti di lavoro secondo le disposizioni del D.Lgs. 81/2008; l) procedimenti di natura disciplinare a carico di studenti; m) valutazioni per la valutazione della didattica e per la customer satisfaction; n) archiviazione e conservazione dati inerenti la carriera universitaria (studi svolti, incarichi ricoperti, titoli di studio conseguiti); o) elezioni rappresentanze studentesche ed eventuale svolgimento di cariche elettive negli													
Titolare del trattamento													
Titolare del trattamento è l'Università degli Studi della Basilicata, nella persona del Rettore pro tempore, con sede a Potenza (85100) - Via Nazario Sauro 85; e-mail: protocollo@pec.unibas.it .													
Responsabile della protezione dei dati													
Dati di contatto del Responsabile della protezione dei dati: rp@unibas.it .													
Finalità e base giuridica del trattamento													
I dati personali, forniti dagli studenti all'atto dell'immatricolazione/iscrizione o durante lo svolgimento della carriera, sono trattati dall'Università degli Studi della Basilicata, nel rispetto dei principi di cui all'art. 5 del GDPR, ai soli fini istituzionali. In particolare, i dati saranno trattati per le seguenti finalità: a) immatricolazione/iscrizione ai corsi di studio e frequenza, sia in presenza sia in modalità e-learning; b) gestione della carriera universitaria; c) gestione dei servizi curricolari ed extracurricolari; d) mobilità nazionale e internazionale; e) determinazione del contributo onnicomprensivo annuale e applicazione delle agevolazioni in materia di tasse; f) conseguimento del titolo di studio; g) utilizzo dei servizi telematici e di posta elettronica; h) accesso ai laboratori e ad altre strutture protette; i) invio comunicazioni inerenti la carriera universitaria; k) applicazione delle misure di sicurezza degli ambienti di lavoro secondo le disposizioni del D.Lgs. 81/2008; l) procedimenti di natura disciplinare a carico di studenti; m) valutazioni per la valutazione della didattica e per la customer satisfaction; n) archiviazione e conservazione dati inerenti la carriera universitaria (studi svolti, incarichi ricoperti, titoli di studio conseguiti); o) elezioni rappresentanze studentesche ed eventuale svolgimento di cariche elettive negli													

Figura 3.10: Lo "stego PDF" con immagini colorate e bianche ottenuto in seguito al processo di Embedding.

3.3.2 Processo di estrazione

Se, al contrario, l'utente decide di **estrarre un messaggio segreto** a partire da un documento PDF, il sistema mostrerà la seguente schermata:

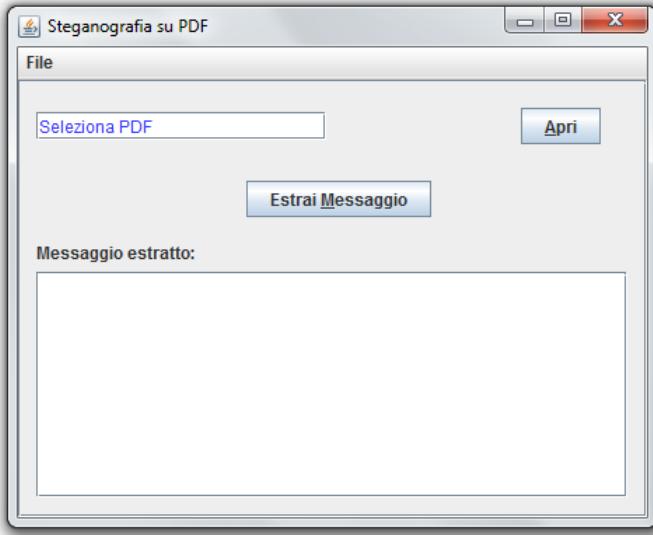


Figura 3.11: Schermata di Estrazione dell'applicazione "Steganografia su PDF."

Supponiamo che l'utente voglia, ora, estrarre un messaggio segreto a partire da uno *stego PDF*. Come abbiamo già illustrato in precedenza, a seconda che il documento abbia solo contenuto testuale oppure abbia sia del testo che immagini, effettuerà l'estrazione del messaggio segreto in maniera differente. Analizziamo separatamente i **2 casi**.

Estrazione da un documento contenente solo del testo

Se il documento PDF contiene solamente del testo, il processo di estrazione effettuerà essenzialmente i seguenti step:

- il sistema estraе le immagini bianche di dimensione 4x4, 8x8, 16x16, 32x32 dallo *stego PDF*;
- per ogni immagine estratta, si inserisce una sottosequenza di due bit corrispondente in una lista di stringhe;
NOTA: *iText* effettua l'estrazione delle immagini dalla prima pagina all'ultima pagina del PDF, ma non lo fa partendo dall'inizio di ciascuna pagina, bensì a partire dalla **fine** della pagina esaminata. Pertanto, una volta estratte le sottosequenze, sarà necessario invertire l'ordine di tale lista: quindi, il primo elemento della lista delle sottosequenze, diventerà l'ultimo, e così via...
- una volta ordinata (in maniera inversa) la lista delle sottosequenze, si costruisce il messaggio binario estratto;
- si converte il messaggio binario estratto;
- alla fine del processo di estrazione, il sistema mostrerà la stringa estratta dallo *stego PDF*.

Di sotto sono riportati alcuni frammenti di codice sorgente Java utilizzati per il processo di Estrazione.

```

1 private void estraiMessaggioPagina(PdfReader reader) throws IOException {
2     for(int i = 1; i <= reader.getXrefSize(); i++) {
3         PdfObject pdfo = reader.getPdfObject(i);
4         if(pdfo != null && pdfo.isStream()){
5             PRStream stream = (PRStream)pdfo;
6             PdfObject type = stream.get(PdfName.SUBTYPE);
7             if(type != null && type.toString().equals(PdfName.IMAGE.toString())){
8                 PdfImageObject pio = new PdfImageObject(stream);
9                 BufferedImage bi = pio.getBufferedImage();
10                if(bi.getHeight() == 4 && bi.getWidth() == 4){
11                    listaSottoseq.add("00");
12                }else if(bi.getHeight() == 8 && bi.getWidth() == 8){
13                    listaSottoseq.add("01");
14                }else if(bi.getHeight() == 16 && bi.getWidth() == 16){
15                    listaSottoseq.add("10");
16                }else if(bi.getHeight() == 32 && bi.getWidth() == 32){
17                    listaSottoseq.add("11");
18                }
19            }
20        }
21    }
22}
23
24
25 private String convertiMessaggio(String messEstratto) {
26     int i = 0;
27     int c = 8;
28     while(messEstratto.length() != 0){
29         String sottos = messEstratto.substring(i, c);
30         String copy = messEstratto;
31         messEstratto = copy.substring(c, copy.length());
32         risultato = risultato + converti(sottos);
33     }
34     return risultato;
35 }
36
37
38 private String converti(String sottos) {
39     String carattere = "";
40     int ascii = Integer.parseInt(sottos, 2);
41     char c = (char)ascii;
42     carattere = carattere + c;
43     return carattere;
44 }
```

NOTA: Dal momento che il processo di embedding inserisce lo stesso messaggio segreto per ogni pagina del documento PDF, il sistema mostrerà solamente una volta il messaggio estratto, anziché mostrare il messaggio estratto tante volte quante sono le pagine del PDF.

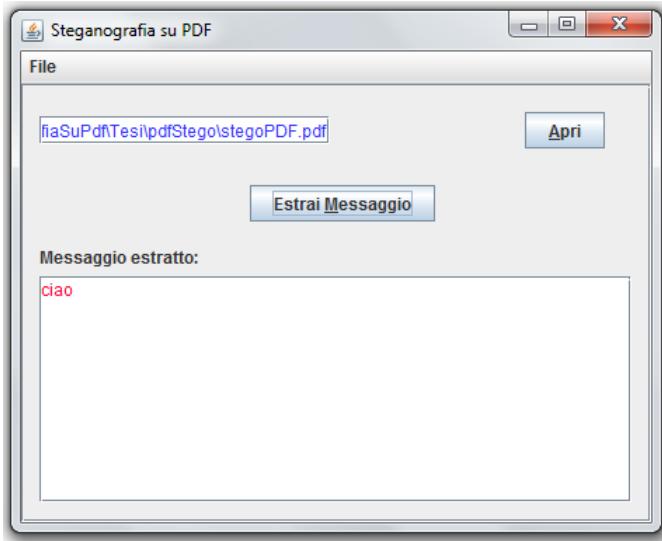


Figura 3.12: L'applicazione "**Steganografia su PDF**" per ogni pagina dello *stego PDF* estrae il messaggio segreto.

Estrazione da un documento contenente sia immagini sia testo

In questo caso il sistema effettuerà l'estrazione in **2 passaggi principali**:

1. estrarrà per ogni pagina del PDF il messaggio nascosto, così come abbiamo illustrato nel caso in cui il documento PDF contenga solamente del testo;
2. alla fine, estrarrà tutte le immagini contenute nel PDF (tranne quelle bianche di dimensione 4x4, 8x8, 16x16, 32x32) e per ciascuna immagine estratta effettuerà l'estrazione del messaggio segreto utilizzando l'**algoritmo F5**.

Capitolo

4

RISULTATI Sperimentali

Il seguente capitolo mostra alcuni "risultati sperimentali", ottenuti eseguendo il processo steganografico illustrato nel **Capitolo 3**.

4.1 Dataset per gli Esperimenti

Per prima cosa forniremo alcuni *risultati*, ottenuti in seguito all'esecuzione dell'algoritmo proposto in questa tesi, prendendo in considerazione un set di **30 documenti PDF**¹: alcuni di essi contengono solo testo, mentre altri contengono sia immagini sia testo. In particolare, di seguito, è riportata una tabella, in cui viene riportato il numero delle pagine del documento PDF, preso in considerazione, e la dimensione del *cover PDF* e dello *stego PDF*.

COVER PDF	NUMERO PAGINE	DIMENSIONE COOVER PDF	DIMENSIONE STEGO PDF
coverImmagini1.pdf	2	60,0 KB	76,0 KB
coverImmagini2.pdf	1	72,0 KB	104 KB
coverImmagini3.pdf	2	124 KB	860 KB
coverImmagini4.pdf	35	600 KB	816 KB
coverImmagini5.pdf	1	184 KB	1,23 MB
coverImmagini6.pdf	3	196 KB	1,07 MB
coverImmagini7.pdf	19	224 KB	532 KB
coverImmagini8.pdf	2	940 KB	1,22 MB
coverImmagini9.pdf	17	5,55 MB	1,22 MB
coverImmagini10.pdf	5	360 KB	412 KB
coverImmagini11.pdf	3	112 KB	124 KB
coverImmagini12.pdf	1	172 KB	1,12 MB
coverImmagini13.pdf	20	352 KB	524 KB
coverTesto1.pdf	3	92,0 KB	112 KB
coverTesto2.pdf	2	16,0 KB	32,0 KB

¹I documenti PDF utilizzati sono stati presi dal sito dell' **UNIBAS**[27].

COVER PDF	NUMERO PAGINE	DIMENSIONE COOVER PDF	DIMENSIONE STEGO PDF
coverTesto3.pdf	1	32,0 KB	48,0 KB
coverTesto4.pdf	27	580 KB	988 KB
coverTesto5.pdf	16	708 KB	740 KB
coverTesto6.pdf	8	120 KB	152 KB
coverTesto7.pdf	15	152 KB	200 KB
coverTesto8.pdf	16	324 KB	372 KB
coverTesto9.pdf	1	8,00 KB	24,0 KB
coverTesto10.pdf	1	12,0 KB	28,0 KB
coverTesto11.pdf	1	8,00 KB	24,0 KB
coverTesto12.pdf	2	16,0 KB	36,0 KB
coverTesto13.pdf	1	8,00 KB	24,0 KB
coverTesto14.pdf	12	64,0 KB	108 KB
coverTesto15.pdf	9	228 KB	260 KB
coverTesto16.pdf	9	228 KB	260 KB
coverTesto17.pdf	2	72,0 KB	124 KB

Figura 4.1: Alcuni risultati sperimentali, ottenuti in seguito all'esecuzione del **processo steganografico**, ivi illustrato.

Per i seguenti risultati sono stati inseriti lo stesso messaggio steganografico ("ciao") e la stessa password ("pass"); ciò che cambia è solamente il *cover PDF*, sui cui è stato eseguito il *processo di embedding*.

4.2 Caratteristiche dello Stego PDF

Abbiamo già notato nel **Capitolo 3** che l'inserimento del messaggio segreto attraverso delle **immagini bianche** viene effettuato da *iText* sottponendolo al contenuto originale del documento PDF, in modo tale che esso non venga oscurato.

L'esempio, riportato di seguito, mostra il contenuto di uno *stego PDF*, all'interno del quale è presente un'informazione steganografica, il cui *processo di embedding* è avvenuto attraverso l'inserimento di **immagini colorate**, anziché bianche proprio per evidenziare questa caratteristica. Per questo esempio, si è scelta come informazione steganografica la stringa "rocchina romano" e come password la stringa "pass".



1. PREMESSA

La presente relazione, predisposta ai sensi dell'art. 14, comma 3, lett. a), del vigente Regolamento di Ateneo per l'amministrazione, la finanza e la contabilità, illustra il contenuto del bilancio unico di previsione dell'esercizio 2016 e i criteri adottati nella sua redazione.

Al fine di fornire un quadro conoscitivo esauriente sul processo di definizione della proposta di bilancio sottoposta all'esame del Consiglio di Amministrazione, si ritiene opportuno richiamare la normativa di riferimento in materia di contabilità delle università statali e svolgere alcune considerazioni preliminari tese ad evidenziare la complessità e le criticità che hanno caratterizzato, sul piano tecnico, i lavori di predisposizione dei documenti previsionali.

In particolare, con riferimento all'ordinamento contabile delle università statali, si rammenta che:

- ai sensi dell'art. 5, comma 1, lettera b) e dell'art. 4, lettera a) della legge 30/12/2010 n. 240 (*c.d. legge di Riforma Gelmini*), il Governo è stato delegato ad adottare uno o più decreti legislativi finalizzati al raggiungimento dei seguenti obiettivi:
 - a) revisione della disciplina concernente la contabilità, al fine di garantirne coerenza con la programmazione triennale di ateneo, maggiore trasparenza ed omogeneità, e di consentire l'individuazione della esatta condizione patrimoniale dell'ateneo e dell'andamento complessivo della gestione;
 - b) introduzione di un sistema di contabilità economico-patrimoniale e analitica, del bilancio unico e del bilancio consolidato di ateneo sulla base di principi contabili e schemi di bilancio stabiliti e aggiornati dal Ministero, di

Figura 4.2: Esempio di *stego PDF* con immagini colorate.

Se l'utente decide di passare il mouse sopra il contenuto steganografico, è possibile osservare che il cursore cambia, evidenziandone la presenza.

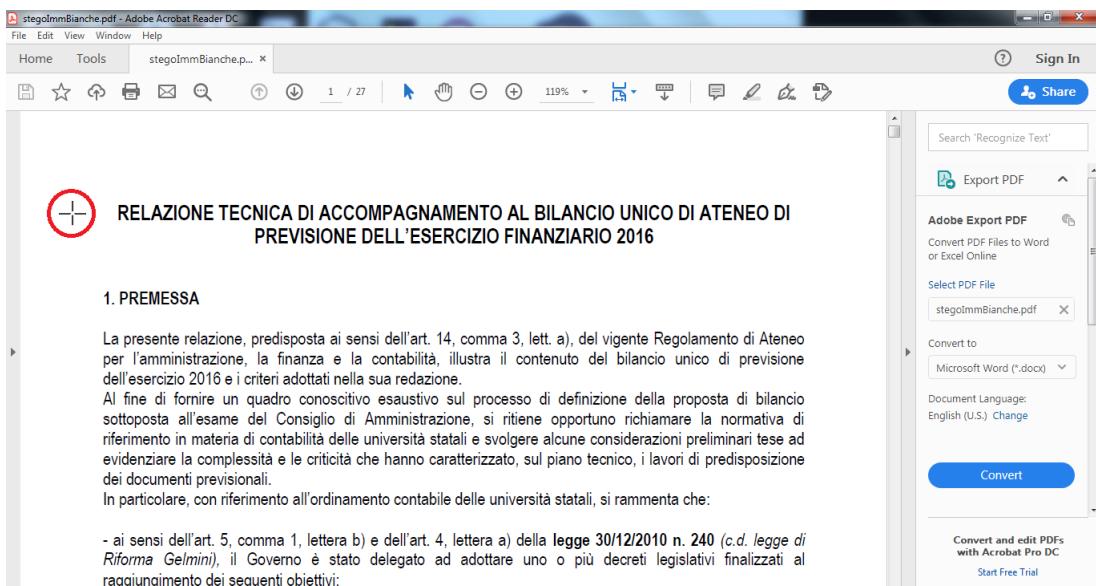


Figura 4.3: Il cursore (cerchiato in rosso) evidenzia la presenza di un'immagine.

4.3 Contenuto Copiato in un Documento MS-Word

Supponiamo di copiare il contenuto di uno *stego PDF*, ottenuto utilizzando il *processo steganografico*, descritto in questa Tesi, in un **documento MS-Word**. E' possibile osservare che il contenuto steganografico non **sopravvive** al "copia ed incolla" del contenuto, come è evidente dall'esempio riportato di sotto.

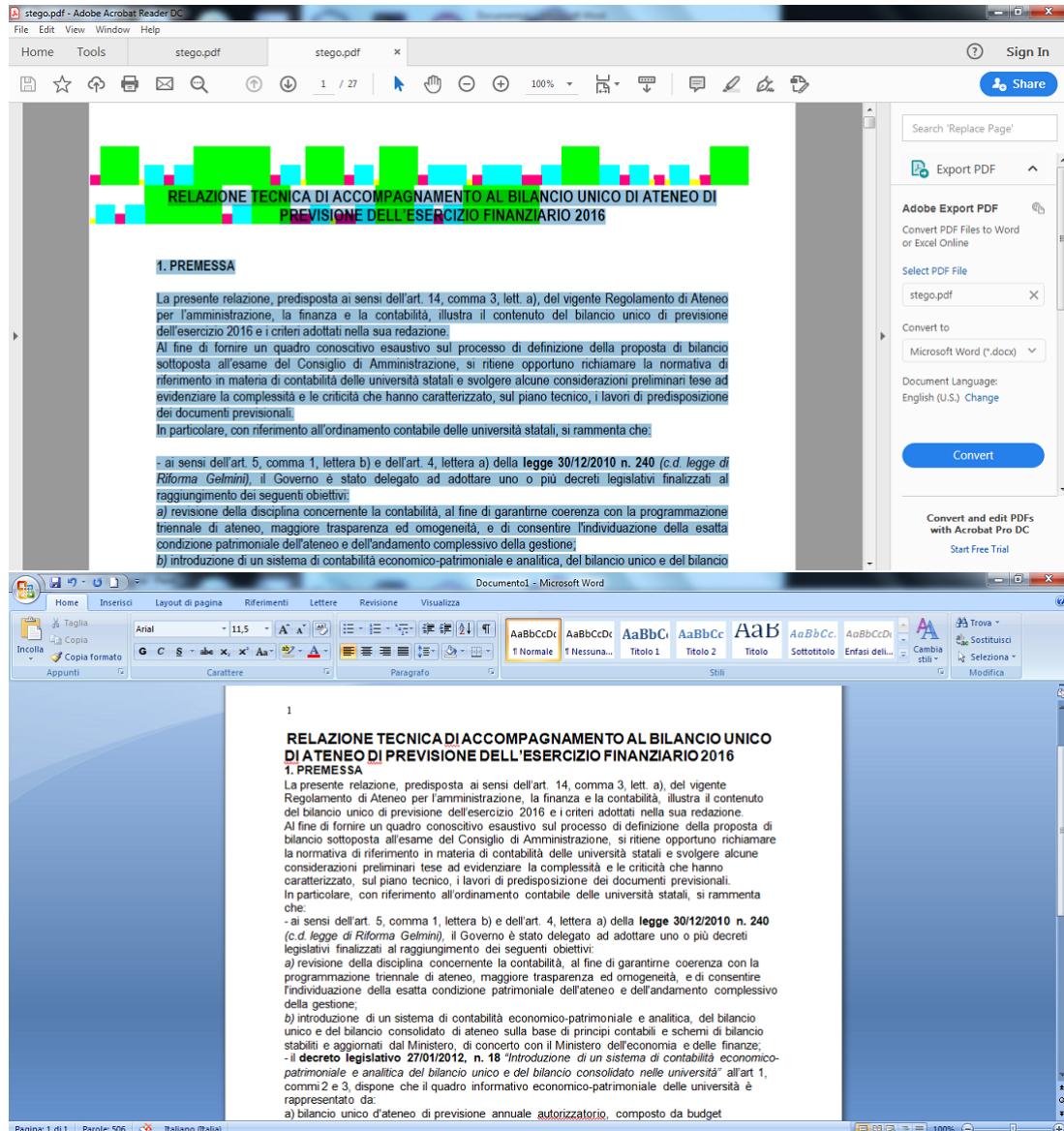


Figura 4.4: Contenuto di uno *stego PDF* copiato in un documento *MS-Word*.

4.4 Unione di più Stego PDF

Vediamo, ora, cosa accade se decidessimo di unire 2 *stego PDF* contenenti informazioni steganografiche diverse, il cui *Processo di Embedding* è stato effettuato inserendo lo stesso *messaggio segreto* per tutte le pagine del PDF. In particolare, in uno si è scelto di inserire la stringa "*rocchina romano*", mentre nel secondo la stringa "*steganografia*". Come è possibile osservare, entrambe le informazioni steganografiche sopravvivono all'unione dei due documenti PDF².

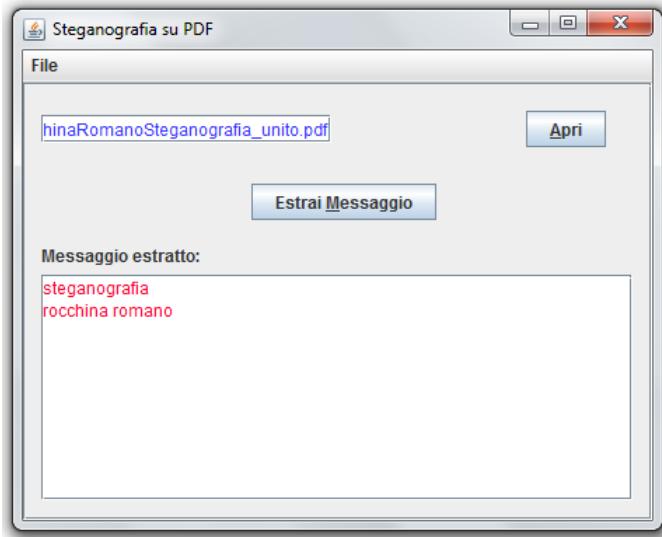


Figura 4.5: Estrazione del contenuto steganografico in seguito all'unione di due *stego PDF*.

²Il sito utilizzato per l'unione dei due *stego PDF* è il seguente [28].

4.5 Ritaglio di un File PDF

Supponiamo di effettuare l'embedding del messaggio segreto "*rocchina romano*" e di dividere la corrispondente stringa del messaggio segreto in forma binaria per il numero di pagine del PDF. In seguito al processo di estrazione, il sistema mostrerà il seguente messaggio estratto:

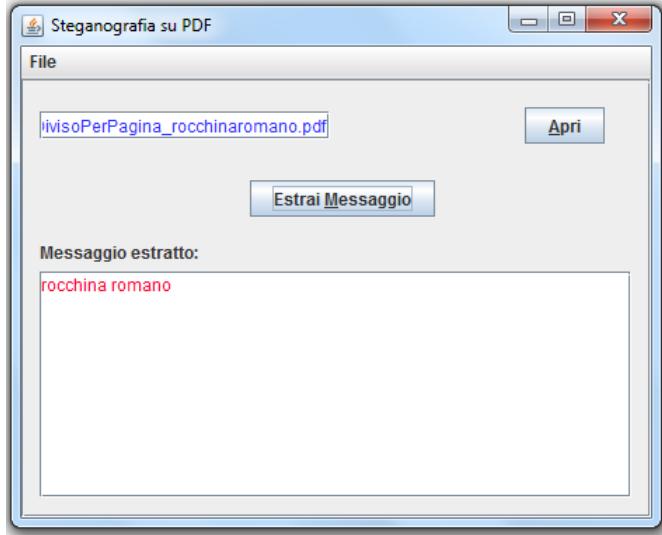


Figura 4.6: Estrazione di un messaggio segreto, il cui contenuto durante il processo di embedding è stato diviso per le pagine del documento PDF.

Vediamo cosa accade se decidessimo di eliminare alcune pagine dello *stego PDF*³. Come è possibile osservare, il sistema ci restituirà un **messaggio di errore**, in cui ci avvisa che il contenuto steganografico, presente nello *stego PDF*, è **corrotto**.

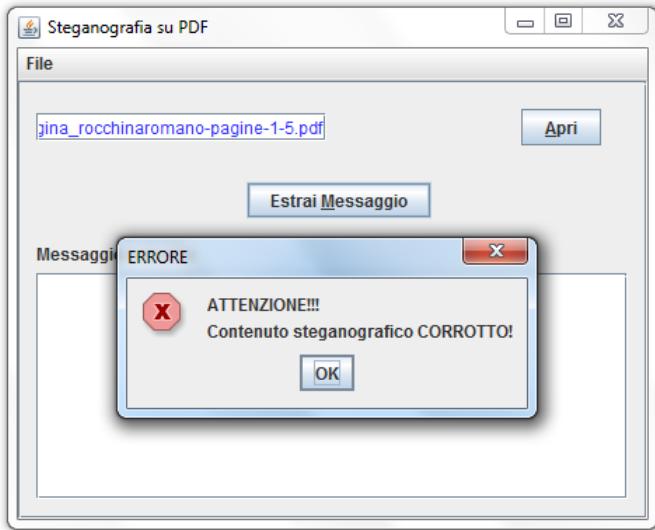


Figura 4.7: Contenuto steganografico corrotto in seguito all'eliminazione di alcune pagine dello *stego PDF*.

NOTA: lo *stego PDF*, preso in esame, ha 9 pagine ed abbiamo deciso di eliminare le pagine 6, 7, 8, 9. Di seguito è riportato il codice sorgente Java utilizzato.

³Il sito utilizzato per eliminare le pagine dello *stego PDF* è il seguente [28].

```

1  private static void inserisciMessaggio(int numPag) {
2
3     for(int i = 0; i <= listaSottosequenze.size()−1; i++ ){
4         String s = (String)listaSottosequenze.get(i);
5             if(i >= numPag−1){
6                 inserisciUltima(s, numPag, i);
7             }else{
8                 String pathOut = pathTemp + pagina + ".pdf";
9                 inserisci(s, i+1, pathOut);
10                coverAppoggio = pathOut;
11                pagina++;
12            }
13        }
14    }
15
16 private static void inserisci(String s, int numPag, String output) {
17     try {
18         float x = 1;
19         float y = 773;
20         PdfReader reader = new PdfReader(coverAppoggio);
21         PdfStamper stamper = new PdfStamper(reader, new FileOutputStream(output));
22         if(s.equals("00")){
23             Image i1 = Image.getInstance(immagine1);
24             i1.setAbsolutePosition(x, y);
25             stamper.getUnderContent(numPag).addImage(i1);
26         }else if(s.equals("01")){
27             Image i2 = Image.getInstance(immagine2);
28             i2.setAbsolutePosition(x, y);
29             stamper.getUnderContent(numPag).addImage(i2);
30         }else if(s.equals("10")){
31             Image i3 = Image.getInstance(immagine3);
32             i3.setAbsolutePosition(x, y);
33             stamper.getUnderContent(numPag).addImage(i3);
34         }else if(s.equals("11")){
35             Image i4 = Image.getInstance(immagine4);
36             i4.setAbsolutePosition(x, y);
37             stamper.getUnderContent(numPag).addImage(i4);
38         }
39         stamper.close();
40         reader.close();
41     } catch (IOException | DocumentException ex) {
42         System.err.println(ex.getMessage());
43     }
44 }

```

```

1 private static void inserisciUltima(String s, int numPag, int i) {
2     try {
3         String pathOut = "";
4             if(i == listaSottosequenze.size() - 1){
5                 pathOut = stego;
6             }else{
7                 pathOut = pathTemp + pagina + ".pdf";
8                 pagina++;
9             }
10            PdfReader reader = new PdfReader(coverAppoggio);
11            PdfStamper stamper = new PdfStamper(reader, new FileOutputStream(pathOut));
12            if(s.equals("00")){
13                Image i1 = Image.getInstance(immagine1);
14                i1.setAbsolutePosition(x1, y1);
15                stamper.getUnderContent(numPag).addImage(i1);
16                x1 = x1 + (float)0.5 + i1.getWidth();
17            }else if(s.equals("01")){
18                Image i2 = Image.getInstance(immagine2);
19                i2.setAbsolutePosition(x1, y1);
20                stamper.getUnderContent(numPag).addImage(i2);
21                x1 = x1 + (float)0.5 + i2.getWidth();
22            }else if(s.equals("10")){
23                Image i3 = Image.getInstance(immagine3);
24                i3.setAbsolutePosition(x1, y1);
25                stamper.getUnderContent(numPag).addImage(i3);
26                x1 = x1 + (float)0.5 + i3.getWidth();
27            }else if(s.equals("11")){
28                Image i4 = Image.getInstance(immagine4);
29                i4.setAbsolutePosition(x1, y1);
30                stamper.getUnderContent(numPag).addImage(i4);
31                x1 = x1 + (float)0.5 + i4.getWidth();
32            }
33            coverAppoggio = pathOut;
34            stamper.close();
35            reader.close();
36            if(x1 > 550){
37                x1 = 1;
38                y1 = y1 - 32 - 1;
39            }
40        } catch (IOException | DocumentException ex) {
41            System.err.println(ex.getMessage());
42        }
43    }

```

Capitolo

5

CONCLUSIONI

Il seguente capitolo riassume l'intero lavoro svolto e propone alcuni sviluppi futuri.

Come abbiamo illustrato nel **Capitolo 3**, l'obiettivo principale di questa Tesi è stato quello di fornire una nuova tecnica steganografica che consentisse di inserire e di estrarre un messaggio segreto partendo da un documento PDF ("Portable Document Format"). In particolare, a seconda che un documento PDF contenga testo oppure testo ed immagini, il processo di embedding e di estrazione effettueranno degli step differenti.

Processo di embedding e di estrazione quando il documento PDF contiene solamente testo

Processo di embedding

Il *processo di embedding* effettua essenzialmente i seguenti step:

1. si converte il messaggio segreto nella corrispondente stringa binaria;
2. si suddivide la stringa binaria in sottosequenze di 2 bit, facendo uso di una lista di sottosequenze;
3. per ogni elemento della lista delle sottosequenze, inizializzata al passo precedente, si inserisce per ogni pagina del documento PDF un'immagine bianca, utilizzando il seguente **schema di codifica**:
 - se la sottosequenza è **00**, la sostituiremo con un'**immagine bianca** di dimensione **4x4**;
 - se la sottosequenza è **01**, la sostituiremo con un'**immagine bianca** di dimensione **8x8**;
 - se la sottosequenza è **10**, la sostituiremo con un'**immagine bianca** di dimensione **16x16**;
 - se la sottosequenza è **11**, la sostituiremo con un'**immagine bianca** di dimensione **32x32**.

NOTA: per la conversione in binario si è considerata la tabella ASCII; pertanto, ciascun carattere è una sequenza di 8 bit.

Processo di estrazione

Per quanto riguarda *il processo di estrazione*, gli step da seguire sono i seguenti:

1. si estraggono le immagini bianche di dimensione 4x4, 8x8, 16x16, 32x32;
2. in base alla dimensione dell'immagine bianca estratta si effettuano le seguenti sostituzioni:
 - se l'immagine bianca ha dimensione **4x4**, si sostituisce con una sequenza di 2 bit pari a **00**;
 - se l'immagine bianca ha dimensione **8x8**, si sostituisce con una sequenza di 2 bit pari a **01**;
 - se l'immagine bianca ha dimensione **16x16**, si sostituisce con una sequenza di 2 bit pari a **10**;
 - se l'immagine bianca ha dimensione **32x32**, si sostituisce con una sequenza di 2 bit pari a **11**.
3. una volta attribuita a ciascuna immagine la corrispondente sequenza binaria di 2 bit, si ottiene il messaggio binario contenuto nel documento PDF;
4. si converte il messaggio binario secondo la tabella ASCII, considerando il fatto che ciascun carattere è una sequenza di 8 bit; di conseguenza, il messaggio binario estratto sarà suddiviso in sottosequenze di 8 bit, ciascuna delle quali corrisponderà ad un determinato carattere nella tabella ASCII.

Processo di embedding e di estrazione quando il documento PDF contiene sia testo sia immagini

Processo di embedding

Il *processo di embedding* effettua essenzialmente i seguenti step:

- si estraggono le immagini contenute nel documento PDF;
- il messaggio segreto viene inserito nelle immagini contenute nel PDF, utilizzando l'algoritmo F5, descritto nel **Capitolo 2**;
- infine, si effettua l'inserimento del messaggio segreto, eseguendo gli stessi step visti quando il PDF contiene solo del contenuto testuale.

Processo di estrazione

Per quanto riguarda *il processo di estrazione*, gli step da seguire sono i seguenti:

- Si estraggono le immagini presenti nel documento PDF, tranne le immagini bianche di dimensioni 4x4, 8x8, 16x16, 32x32;
- si procede con l'estrazione del messaggio segreto dalle immagini, contenute nel documento PDF, utilizzando l'**algoritmo F5**, descritto nel **Capitolo 2**;
- infine, si estra il messaggio segreto, eseguendo gli stessi step visti quando il PDF contiene solo del contenuto testuale.

5.1 Risultati Raggiunti

Nel **Capitolo 4** sono stati mostrati alcuni risultati sperimentali, ottenuti eseguendo il *processo di embedding* ed il *processo di estrazione*, proposti in questa Tesi.

Per poter realizzare un processo steganografico di questo tipo, è stata realizzata un'applicazione con interfaccia grafica, scritta in **Java**, sfruttando la libreria **iText**, che offre una serie di metodi per la lettura, la scrittura e la modifica di documenti in formato PDF.

I risultati raggiunti sono stati i seguenti:

- *iText* permette di inserire le immagini bianche sottoponendole al contenuto originale del documento PDF in modo tale da non oscurarlo;
- le immagini bianche inserite nello *stego PDF* vengono rilevate dal mouse;
- il contenuto steganografico non "sopravvive", se decidessimo di copiare parte dello *stego PDF* in un documento MS-Word;
- il contenuto steganografico, invece, "sopravvive", se uniamo due o più *stego PDF*;
- Se eliminiamo alcune pagine dello *stego PDF*, invece, è possibile osservare che
 1. se il messaggio segreto inserito è uguale per tutte le pagine del documento PDF, il contenuto steganografico "sopravvive";
 2. al contrario, se durante il processo di embedding il messaggio segreto è stato diviso per il numero di pagine del documento PDF, il contenuto steganografico andrebbe perso.

5.2 Codice Sorgente ed Utilizzo dell'Applicazione

L'applicazione e, quindi, il *codice sorgente Java*, relativo ad essa, è disponibile su <https://github.com/rocchinaRomano?tab=repositories>, nella repository "**steganografiaSuPDF**", all'interno della quale è possibile trovare il file "**README.txt**", nel quale vi sono tutte le informazioni necessarie per il download e l'utilizzo dell'applicazione.

NOTA: Per poter scaricare ed utilizzare l'applicazione, è necessario installare la *Java Virtual Machine*, disponibile su <https://www.java.com/it/>.

5.3 Sviluppi Futuri

In questo paragrafo cercheremo di fornire alcuni dei possibili *sviluppi futuri*, riguardanti il processo steganografico proposto in questa Tesi:

- migliorare il **tempo di esecuzione** per il *processo di embedding*.
- implementare il processo steganografico, ivi proposto, utilizzando un'**altra libreria** anche in un linguaggio di programmazione diverso da Java;
- per evitare che il mouse possa rilevare la presenza del contenuto steganografico, si potrebbe individuare l'area del documento PDF, in cui è inserito il testo, e fornirla come prima coordinata per l'inserimento delle immagini bianche;
- sviluppare un'applicazione per **dispositivi mobile** (*iOS* o *Android*);
- cambiare il **processo di codifica/decodifica** per l'inserimento e l'estrazione del messaggio segreto;
- trovare un possibile **attacco di steganalisi** per questo processo steganografico.

BIBLIOGRAFIA

- [1] *Il codice della Gioconda e il mistero del ponte del diavolo*, 2010. <https://forum.termometropolitico.it/97365-la-gioconda-e-il-ponte-gobbo-sul-trebbia.html>.
- [2] *Opere d'arte insolite. Opere d'arte misteriose*, 2012. <https://antveral.wordpress.com/2012/06/10/3143/>
- [3] R. E. Saputra, A. Febryan, T. W. Purboyo. *Steganography methods on text, audio, image and video: A survey.*, 2017.
- [4] V. Sundaram, A. Joseph. *Cryptography and steganography – a survey.* (Vol.2(3)), 2011.
- [5] M. Faezipour, A. Odeh, K. Elleithy. *Steganography in text by using MS word symbols.*, 2014.
- [6] M. Agarwal. *Text steganographic approaches:a comparison.*, 2013.
- [7] J. Huang, Y. Q. Shi, B. Li, J. He. *A survey on image steganography and steganalysis.* (2), 2011. Journal of Information Hiding and Multimedia Signal Processing.
- [8] T. Battini. *Intervista - Silvano Vinceti: "Il vero 'codice' Da Vinci? È negli occhi della Gioconda.*, 2011. <https://www.nannimagazine.it/articolo/7353/intervista-silvano-vinceti-il-vero-codice-da-vinci-È-negli-occhi-della-gioconda>.
- [9] B.Ramapriya. *An improved approach of text steganography in application with rotational symmetry.*, 2017.
- [10] M. Kutter, F. Hartung. *Multimedia watermarking techniques.*, 1999.
- [11] K. Saroha, H. Singh, P. K. Singh. *A survey on text based steganography.*, 2009.
- [12] S. Upadhyaya, H. Singh, A. Diwakar. *A novel approach to text steganography.*, 2014.
- [13] K. W.Ross, J. F.Kurose. *Reti di calcolatori e Internet. Un approccio top-down.* Quarta edizione.
- [14] J. R. Meehan, T. Bienz, R. Cohn. *Portable Document Format. Reference Manual.*, 1997.
- [15] D. Hoga, J. Fridrich, M. Goljan. *Steganalysis of jpeg images: Breaking the f5 algorithm.*, 2002.
- [16] J.R.Krenn. *Steganography and steganalysis.*, 2004.

- [17] V. Lavecchia. *Differenza tra crittografia e steganografia in sicurezza informatica.* <https://vitolavecchia.altervista.org/differenza-tra-crittografia-e-steganografia-in-sicurezza-informatica/>.
- [18] Wikipedia. *Portable Document Format (PDF).*
- [19] G. Montemarani. *Steganografia. L'arte della scrittura nascosta.*, 2005.
- [20] P. Honeyman, N. Provos. *Hide and seek: an introduction to steganography.*, 2003.
- [21] S. Kumar, P. Reddy. *Steganalysis techniques: a comparative study.*, 2007.
- [22] M. R. Mokhtar, R. Sulaiman, S. S. Baawi. *New text steganography technique based on a set of two-letter words.*, 2017.
- [23] T. Chen, S. Zhong, X. Cheng. *Data hiding in a kind of pdf texts for secret communication.*, 2007.
- [24] D. AL-Nasrawi. W. Bhaya, A. M. Rahma. *Text steganography based on font type in MS-word documents.*, 2013.
- [25] A. Westfeld. *F5-A steganographic algorithm. High capacity despite better steganalysis.*, 2001.
- [26] iText: <https://itextpdf.com/en>.
- [27] Portale UNIBAS: <http://portale.unibas.it/site/home.html>.
- [28] Smallpdf: <https://smallpdf.com/it/unire-pdf>.