

UNIVERSITÀ DEGLI STUDI DI CATANIA

DIPARTIMENTO DI MATEMATICA E INFORMATICA

Corso di Laurea Magistrale in Informatica

*Diego Rando (1000027105)*

**Steganography – Hide an audio into an image**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Relazione Progetto – Multimedia e laboratorio

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Professori:

Prof. Filippo Stanco

Prof. Dario Allegra

Anno Accademico 2020 – 2021

**Indice**

Indice 2

Introduzione 3

Tecniche utilizzate e Pipeline 6

Linguaggio e librerie utilizzate 10

Possibili miglioramenti 11

**INTRODUZIONE**La nostra epoca, basata sulla comunicazione, ha dato ai dati un valore piu ampio di ciò che essi avevano negli anni passati. Si ha spesso bisogno di salvaguardare questi dati e dunque necessitano tecniche in grado di mantenere i dati riservati anche quando essi finiscano in mani di terzi.

La scienza che si occupa di questo problema è la **Crittografia** che ha come obiettivo principale il garantire la segretezza attraverso la codifica del messaggio. Il messaggio è visibile, ma codificato mediante appositi algoritmi di cifratura che lo rendono incomprensibile a chi non è a conoscenza dei relativi metodi di decodifica.

Un’altra tecnica utilizzata è il **Watermarking**, tecnica che permette la possibilità di inserimento di informazioni in un segnale.  Questo segnale deve essere visibile solamente in determinate condizioni, per esempio dopo l’applicazione di opportuni algoritmi.

La **Steganografia** è l’insieme di tecniche che consentono di nascondere messaggi, inserendoli all’interno di un contesto che funge da contenitore ed in grado di nascondere il contenuto e la sua esistenza, agli occhi dell’osservatore.

A differenza della crittografia, l’obiettivo della steganografia è nascondere  
l’esistenza stessa della comunicazione, nascondendo il vero messaggio all’interno di un segnale (audio, testo, immagine, … ) dal significato innocuo.

All’interno del seguente elaborato, utilizziamo algoritmi di steganografia inserendo all’interno della seguente immagine cover:

Immagine che contiene cielo, esterni, via, strada

Descrizione generata automaticamente

il seguente payload, corrispondente ad un segnale audio, dopo essere stato trasformato in immagine:

Immagine che contiene testo

Descrizione generata automaticamente

Il risultato finale viene chiamato stego-image ed è rappresentato dalla seguente immagine:

Immagine che contiene cielo, esterni, via, strada

Descrizione generata automaticamente

Dal risultato finale ottenuto, si evince che non è possibile dire che all’interno della stego-image è presente un messaggio audio.

C’è un notevole incremento delle dimensioni dell’immagine che passa da 328Kb non steganografata ad un 1.2Mb steganografata, con un audio inserito di dimensioni 13.2Mb.

**Tecniche utilizzate e Pipeline**

All’interno del seguente progetto è stata utilizzata una tecnica di steganografia iniettiva in cui abbiamo inserito un payload, rappresentante una nota audio, all’interno dell’immagine cover.

E’ stata impiegata una tecnica di steganografia detta sostitutiva, ovvero una tecnica che consiste nel sostituire i bit meno significativi (LSB least significa bit) dell’immagine con i bit del payload. Considerando che vengono modificati i piani di bit meno significativi è chiaro che l’immagine a livello estetico subirà delle variazioni minime se paragonata con l’immagine originale.

La pipeline svolta è stata la seguente:

1. Viene trasformato il file audio in formato “.mp3” (nel nostro caso è stato trasformato il file audio dal formato ‘.wav’ di partenza). Successivamente viene trasformato in immagine (in formato ‘.png’) tramite la funzione audioToImage(). Si preferisce un formato png in quanto esso è lossless e quindi durante la lavorazione dell’immagine non abbiamo perdita di dettagli.
2. È stata inserita l’immagine del payload all’interno dell’immagine contenitore, tramite la funzione encode() in cui al suo interno è presenta la funzione encode\_enc() che va ad inserire i pixel dell’immagine audio tramite la funzione putPixel() che prende come parametro il pixel e le coordinate x,y di esso stesso e lo inserisce all’interno dell’immagine contenitore.
3. I pixel prima di essere inseriti all’interno dell’immagine vengono trasformati nel formato a 8-bit binario ed inseriti nelle posizione di indice x,y.

È possibile fare il passaggio inverso, tramite la funzione decode(), in cui passiamo un’immagine contenitore, dove è presente un qualche payload al suo interno, e otteniamo sia l’immagine relativa al payload inserito, sia l’immagine originale senza payload.

Alla seguente immagine contenitore:



È stato inserito il seguente payload:

Immagine che contiene testo

Descrizione generata automaticamente

E abbiamo ottenuto il seguente risultato, del tutto “simile” all’immagine contenitore nonostante essa contenga il messaggio steganografato:

Immagine che contiene cielo, esterni, via, scena

Descrizione generata automaticamente

Un altro esperimento è stato effettuato con la seguente immagine contenitore:

**Immagine che contiene cielo, esterni, scena, via

Descrizione generata automaticamente**

A cui è stato inserito il seguente payload:

Immagine che contiene testo

Descrizione generata automaticamente

Per arrivare in fine al seguente risultato:

**Immagine che contiene cielo, esterni, scena, via

Descrizione generata automaticamente**

**Linguaggio e Librerie utilizzate**

È stato utilizzato il linguaggio di programmazione Python e alcune librerie offerte per la manipolazione delle immagini, manipolazione di strutture dati come array e manipolazione dei segnali digitali sotto forma di file audio e immagini.

Una tra queste librerie è PIL che ci ha permesso di modificare a basso livello i pixel dell’immagine.

È stato necessario anche l’utilizzo della libreria numpy, utilizzata per creare array ed inserire al loro interno i dati relativi all’immagine payload.

**Possibili miglioramenti**

Al seguente progetto potrebbero essere applicati dei miglioramenti in termini di robustezza. Potrebbero essere resi più robusti gli algoritmi andando a considerare una possibile modifica dell’immagine steganografata, come per esempio l’applicazione di un filtro.

Per l’attuale algoritmo una modifica di questo genere non ci permetterebbe di ottenere nuovamente i dati originali, a causa della modifica effettuata dal filtro e della poca robustezza garantita dalla tecnica sopra descritta.