Relazione Crittografia Visuale

A cura di Valerio Egidi

Professore: Stefano Bistarelli

La crittografia visuale è una tecnica di "*camuffamento*" di un messaggio tramite immagini che ad occhio nudo sembrano prive di senso o con un significato diverso da quello che rappresentano veramente.

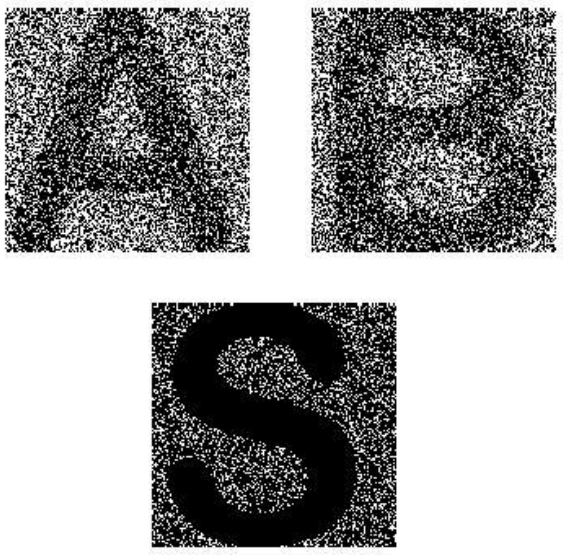
Prima di inoltrarci però oltre sulla definizione vera e propria è necessario esaminare meglio lo stesso nome "*Crittografia Visuale*".

Il termine Crittografia viene dalle parole "*nascosto*" e "*scrittura*", stà quindi ad indicare un concetto molto diffuso nell'ambito della sicurezza informatica e non, significa quindi riuscire a comunicare un messaggio tra due interlocutori senza che un eventuale terzo in ascolto fra i due riesca a coglierne il significato.

Per nascondere il messaggio esistono varie tecniche, quella esaminata da questo piccolo seminario è appunto basata sul riconoscimento "*visuale*" del messaggio. Il mezzo di decodifica infatti non è un algoritmo o un qualsiasi procedimento macchinoso per cui serve l'ausilio di un calcolatore, ma il semplice occhio umano.

Tale tecnica consiste nel comunicare un messaggio sotto forma di immagine. Per "*crittografare*" l'immagine bisogna scomporla in almeno due sotto immagini che vengono chiamati "**share**". La sovrapposizione delle due immagini, effettuando **la somma dei colori**, darà luogo ad una terza immagine che sarà quindi il messaggio originale. Purtroppo tale tecnica non permette di lasciare il messaggio originale totalmente inalterato, infatti sarà caratterizzato da un "*rumore*" di fondo, ovvero, nella pratica, dal classico effetto che abbiamo quando una televisione analogica non trova un segnale video ad una determinata frequenza.

Per rendere meglio l'idea la *Figura 1*. sarà in grado di rendere chiaro tale concetto.



Come possiamo notare i due share originali sembrano rappresentare una "*A*" ed una "*B*". Sommando le due immagini verrà visualizzato il **messaggio segreto** "*S*". Senza un animazione o una prova pratica tramite programmi di grafica può risultare difficile immaginare come sia possibile, ma il meccanismo è più semplice di quanto si possa credere.

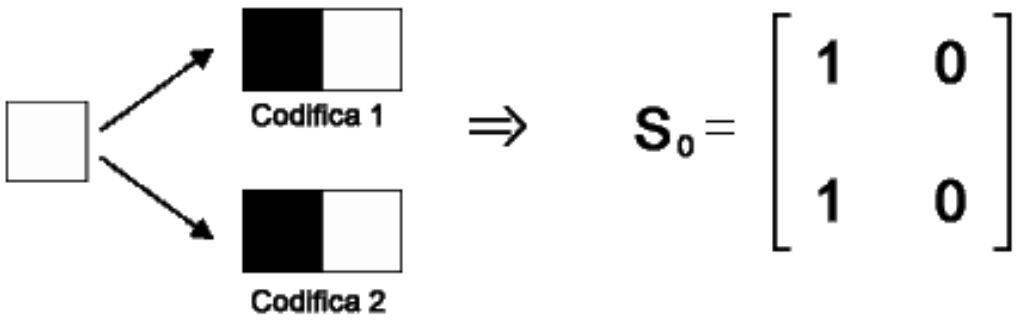
Iniziamo da un approccio pratico a quello che succede con l'operazione della somma fra immagini. Per capirlo a fondo dobbiamo intendere tale somma **pixel per pixel**, ovvero sommeremo il pixel alle coordinate 1,1 della prima immagine con il pixel alle coordinate 1,1 della seconda. Lavorando con immagini a due colori i risultati di tale somma sono pochi e facilmente intuibili: bianco o nero.

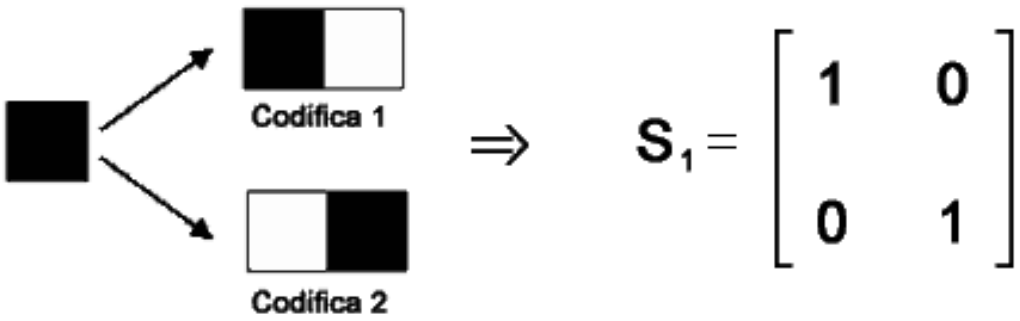
> *Se sommiamo due pixel bianchi la somma dei due risulterà un pixel bianco.*
>
> *Se sommiamo due pixel neri la somma dei due risulterà un pixel nero.*
>
> *Se sommiamo due pixel di colore differente il risultato sarà un pixel nero.*

La somma più importante è proprio la terza, infatti per creare gli share distorti dobbiamo fare leva principalmente su quella somma, mischiando i pixel "*mischiati*" in tutta l'immagine rendendola **irriconoscibile**.

Un eventuale script in grado di creare due *share* partendo da un immagine originale quindi dovrà ragionare in questo modo:

> *Per rappresentare un pixel bianco dovrà scrivere nei due share due pixel bianchi*
>
> *Per rappresentare un pixel nero dovrà scrivere nei due share due pixel neri oppure uno bianco ed uno nero.*





Problema: con questo metodo il messaggio risulterà “sgranato” ma facilmente riconoscibile visto che tutti i pixel bianchi saranno riportati in bianco.

Per risolvere tale problema è necessario introdurre il concetto dei “**subpixel**”. Per *subpixel* si intedono 2 o pixel considerati come “*blocco unico*”, il quale, sommato con un altro *subpixel*, deve essere in grado di rappresentare sia l’informazione di un pixel bianco che di uno nero.

Avendo più pixel a dispozione per rappresentare un bianco o un nero le combinazioni aumentano. Immaginiamo di avere subpixel dalla grandezza di 2 pixel.

Per ottenere un pixel bianco potremo sommare un subpixel *bianco-nero* con un altro *bianco-nero*, oppure uno *nero-bianco* con uno *nero-bianco.*

Sommando due subpixel **uguali** saremo quindi in grado di ottenere come risultato una coppia di pixel, uno bianco ed uno nero, i quali inseriti nel contesto di un immagine realizzata con molti di questi *subpixel* renderanno l’idea di un pixel bianco.

Sommando due *subpixel* **diversi** otterremo una coppia di neri, i quali vogliono quindi comunicare la presenza di un pixel nero.

La somma dei due *share* quindi permetterà di vedere completamente in nero il **messaggio segreto**, mentre il resto risulterà una grana di pixel bianchi e neri apparentemente casuali.

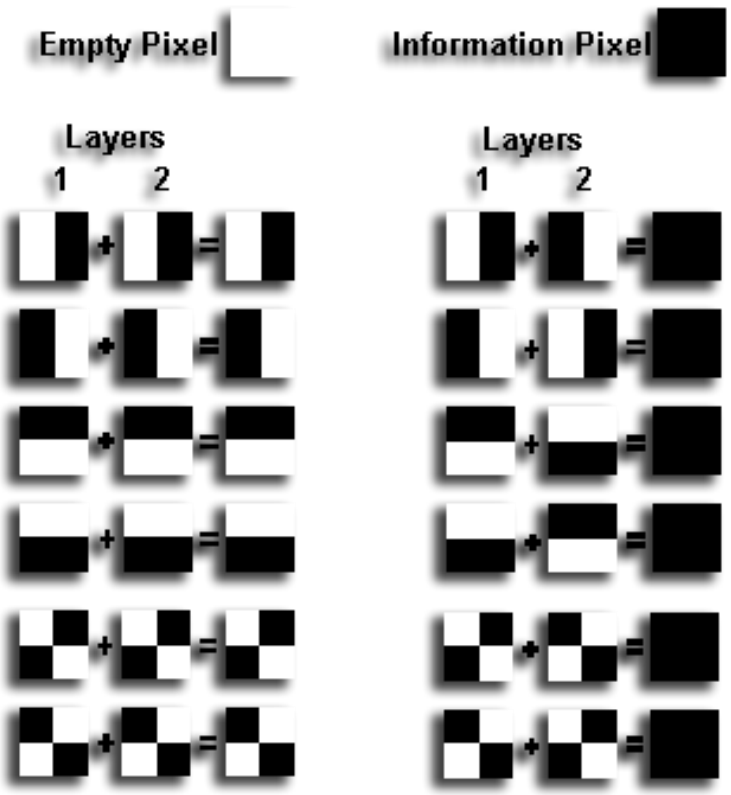
Usare *subpixel* composti da soli due elementi però può risultare svantaggioso, l’immagine finale infatti risulterà distorta oppure con un elevata perdita di informazione. Se i due pixel sono disposti in orizzontale, considerando che servono 2 pixel per rappresentare un singolo pixel originale, l’immagine risulterà il doppio più larga, mentre se si decide di voler utilizzare 2 pixel per rappresentare un pixel originale ed uno “*casuale*” il risultato verrà compromesso.

Risulta quindi utile l’aumento del numero di pixel del *subpixel* a 4.

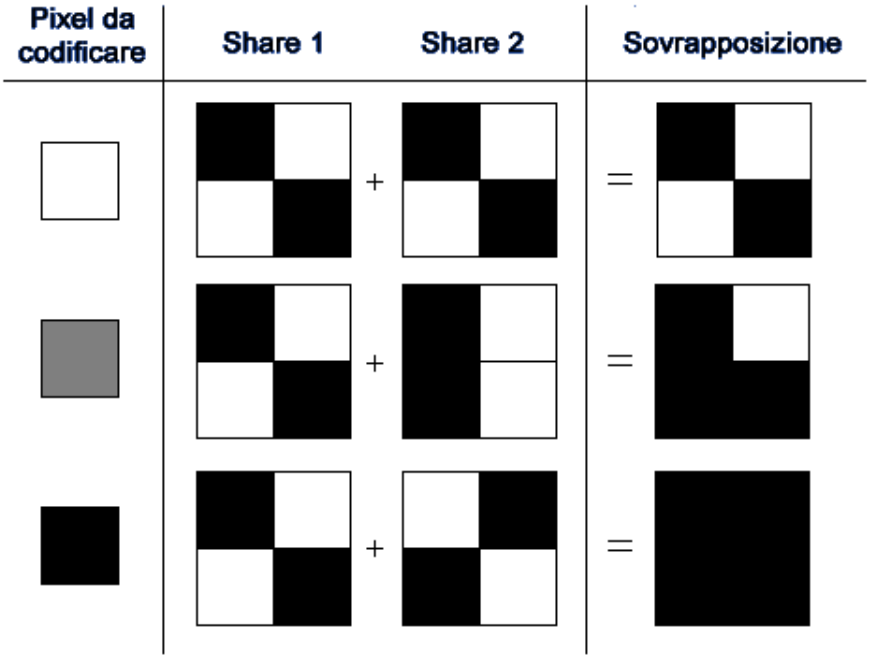
Usando 4 subpixel siamo in grado di creare altre combinazioni di pixel, esattamente 6 combinazioni per rappresentare un pixel bianco ed altre 6 per uno nero.

In questo caso i due share dovranno avere una dimensione in pixel quadrupla rispetto all'immagine originale se non si vuole avere un eccessiva perdita di informazione dal messaggio originario.

Con l'introduzione dei *subpixel* da 4 pixel è anche possibile crittografare un messaggio **a 3 colori**. *Un subpixel formato da 2 pixel bianchi ed uno nero nell'insieme di un immagine può quindi sembrare un pixel bianco, uno formato da 1 pixel bianco e 3 neri può sembrare grigio mentre uno formato da 4 neri sembrerà effettivamente nero.*



L'immagine sottostante fa capire meglio questo concetto.

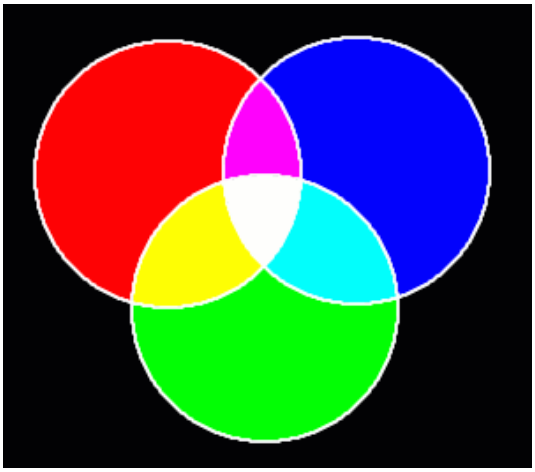


Aumentiamo ancora il numero di colori disponibili, stavolta non basta però aumentare il numero di subpixel visto che il nostro scopo è nascondere il messaggio in un immagine a colori.

Per spiegare come fare bisogna avere delle nozioni di base sulla scomposizione dei colori della luce.

Un fascio di luce che il nostro occhio recepisce come **bianco** è in realtà la somma di **diversi colori**, la somma dei quali lo rende di quel colore. I colori sono semplicemente quelli che siamo abituati a vedere in un arcobaleno, il quale è effettivamente formato dalla scomposizione di un raggio di *luce solare*.

Per creare un *monitor* in grado di rappresentare tutti i colori visibili dall'uomo sarebbe quindi stato necessario che ogni pixel possedesse *tutti quei colori*, la somma dei quali avrebbe ricreato il colore *bianco*. Fortunatamente si scoprì che non servivano tutti questi colori, ma ne bastavano anche 3 sufficentemente "*lontani*" fra di loro all'interno della scala di colore. I 3 colori sono appunto **Rosso Verde e Blu**, meglio conosciuti con la sigla inglese **RGB**. Dosando opportunatamente questi tre colori è infatti possibile rappresentare una vastissima gamma di colori, gli stessi che siamo abituati a vedere nei nostri monitor.

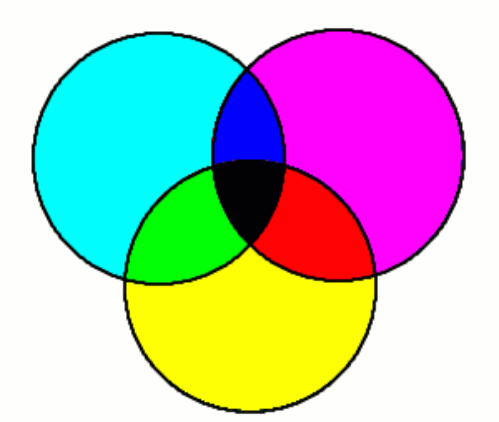


Sommando a coppie questi 3 colori di base (chiamati **colori primari**) è possibile creare altri 3 colori (chiamati **colori secondari o ausiliari**), che sarebbero:

Sommando rosso e verde: Giallo

Sommando Verde e blu: Ciano

Sommando blu e rosso: Magenta



Tramite i *colori ausiliari* è possibile ottenere i *colori primari* tramite un operazione opposta alla somma, chiamata **sintesi sottrattiva**. L'operazione di *sottrazione* non è poi così difficile da immaginare, se ci capita di vedere un foglio giallo sappiamo che è giallo perché, nonostante la luce ambientale sia bianca, tale foglio è in grado di trattenere la colorazione blu, facendo giungere al nostro occhio solamente quelle rossa e verde, le quali una volta sommate ottengono il colore giallo.

Tramite quindi i colori primari siamo in grado di rappresentare con dei numeri i colori a livello di *codice*. Ogni colore primario ha un valore di 8 bit (quindi un numero compreso **fra 0 e 255**) che determina quanto quel colore venga usato.

Per fare degli esempi:

> *Al codice Red = 0, Green = 0, Blue = 0 corrisponde il colore nero.*
>
> *Al codice Red = 255, Green = 255, Blue = 255 corrisponde il colore bianco.*
>
> *Al codice Red = 255, Green = 0, Blue = 0 corrisponde il rosso.*
>
> *Al codice Red = 0; Green = 255, Blue = 255 corrisponde il ciano.*

E' quindi possibile realizzare due share con "circa" **8 colori**, usando *quindi i colori primari, secondari, il bianco ed il nero.*

Consideriamo che vogliamo eseguire la somma fra due colori C1 e C2 formati entrambi dai colori r1, r2 ed r3.

La somma dei due pixel sarà determinata dalla formula: **C1 + C2 = (r1*r1/255, r2*r2/255, r3*r3/255)**

Tale formula **NON** crea un colore "a metà" fra i due che si vanno a sommare, ma effettivamente permette al colore "*dominante*" ti uscirne come risultato.

Se infatti andiamo a sommare un rosa con il nero il risultato sarà sempre nero. Se sommiamo un bianco con un rosso il risultato sarà rosso.

E' quindi possibile nascondere un immagine di 8 colori con le stesse tecniche usate per le immagini a due colori, sostituendo però il colore che vogliamo comunicare alla coppia di bianchi ed usando colori "*a caso*" da sommare ai neri.

Queste sono le tecniche più diffuse per ciò che noi siamo abituati a chiamare *Crittografia Visuale*. A mio avviso però si può includere nell'argomento anche l'uso delle *maschere*.

Le **maschere** sono effettivamente degli algoritmi che permettono di "*scomporre*" l'immagine in modo da renderla irriconoscibile ad occhio nudo ma normale dopo aver riapplicato tali maschere.

Per avere un idea precisa di come funzionano consiglio l'uso del software **gmask** allegato a questo documento, il quale presenta diverse maschere da usare.

La particolarità di queste maschere è data dal fatto che dopo essere stata applicata sia possibile tornare indietro, usando la stessa maschera o usando una maschera che faccia il lavoro opposto della prima.

In questo caso quindi per mandare un messaggio crittografato non basta trasferirlo al nostro destinatario, sarà necessario mandare anche la sequenza di operazioni necessarie per permettere di tornare all'immagine originale.

Presentando in breve le maschere offerte dal software G-mask eccone una semplice descrizione:

***Rotate RGB**: Scambia fra di loro il valore sui canali RGB, assegnando il valore di R a G, G a B, B ad R. Una volta applicata questa maschera è necessario applicarla altre due volte quindi per completare la rotazione e tornare ai colori originali.*

***XOR**: Viene effettuata l'operazione di xor con il numero esadecimale 0x80. L'operatore xor ha la particolarità che venendo eseguito due volte con lo stesso valore il risultato sia lo stesso iniziale. Quindi applicandolo due volte torneremo all'immagine originale.*

***Flip Up-Down/Left-right**: Piuttosto banale, si limita a "specchiare" l'immagine o la frazione di immagine selezionata in verticale o in orizzontale.*

***Vertical Glass Blocks**: Divide l'immagine in colonne, specchiando ogni colonna in orizzontale.*

***Horizontal Glass Blocs**: Divide l'immagine in righe, specchiando ogni riga in verticale.*

***Win**: Divide l'immagine in colonne di "n" pixel, le singole colonne larghe un pixel vengono poi spostate fra di loro, dando un effetto molto confuso all'immagine. Riapplicandola una sola volta si torna all'immagine originale.*

***Meko -**: Questa è forse la più interessante fra le maschere, divide l'immagine in blocchi e li scambia fra di loro. Applicare tale maschera "n" volte non farà altro che continuare a mischiare l'immagine, anche se seguendo un criterio logico. Purtroppo non sono riuscito a trovare informazioni dettagliate su effettivamente quale sia la logica con cui vengono mischiate.*

***Meko +**: Esegue l'operazione contraria del Meko- , infatti applicando "n" volte l'operatore Meko- l'unico modo per tornare all'immagine originaria è applicare lo stesso "n" numero di volte l'operatore Meko+.*

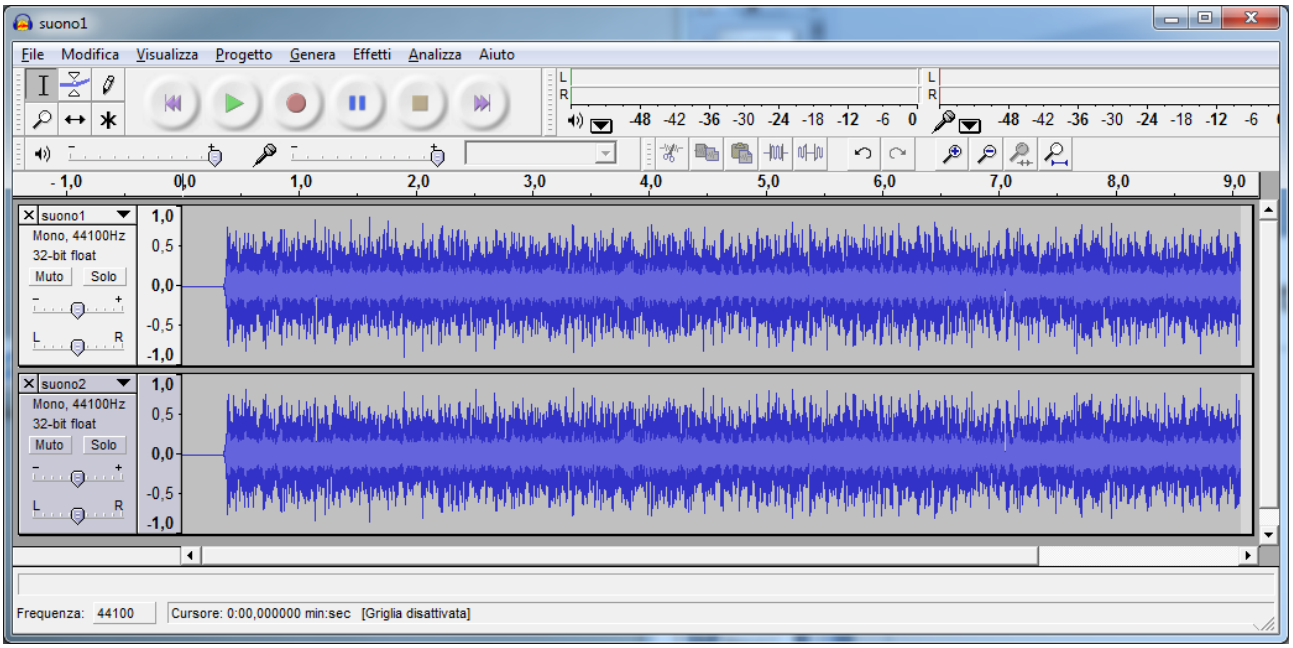
***FL**: Crea il negativo dell'immagine e la riordina a spirale. Riapplicando tale maschera si torna all'immagine originale.*

Curiosità: di tale curiosità non ho realizzato slide visto che si tratta di una cosa che non riguarda strettamente la crittografia Visuale.

L'operazione di somma di immagini è applicabile anche ai suoni, permettendo quindi di nascondere un messaggio audio **altrimenti inudibile**.

L'esempio pratico è nei due file mp3 allegati a questa relazione, sentendoli singolarmente si noterà soltanto un breve pezzo di un brano piuttosto singolare. Ascoltandoli insieme (perfettamente sincronizzati!) si riesce a sentire la voce del sottoscritto in sottofondo che dice "*messaggio segreto*". Per realizzare questa somma è necessario avere delle conoscenze di base delle **onde sonore**. Quando, ad esempio, schiocchiamo le dita produciamo un suono facilmente udibile. Tale suono è creato dalle vibrazioni create dallo sfregamento delle dita che vibrando tramite l'aria arriva alle nostre orecchie. Tale vibrazione è descrivibile quindi come un onda che viaggia a determinate frequenze, nel caso del nostro orecchio su delle frequenze comprese fra i 20 ed i 19000 hz. Il limite di frequenze udibili cambia da persona a persona ed è fortemente influenzato dall'età ma è meglio evitare di divagare oltre.

Essendo il suono un **onda** è possibile quindi creare un suono opposto semplicemente "**ribaltando verticalmente**" tale *onda sonora*. Sovrapponendo le due onde, essendo forze di uguali e contrarie, si annulleranno a vicenda lasciando solamente il **silenzio**.



Per realizzare i due file di esempio ho ritagliato un piccolo pezzo musicale (scegliendo non a caso un pezzo molto confusionario!), convertito in mono, duplicato e "*ribaltato*" la copia. Ho poi aggiunto il messaggio segreto ad uno dei due, fidandomi del fatto che ascoltando normalmente il file la mia voce sarebbe stata coperta dalla musica assordante.

Il funzionamento è *semplice* ed il risultato è *ottimo*. I disturbi di fondo udibili dalla somma dei due file sono dovuti alla compressione *mp3* dei file, infatti tale formato non è senza perdita di informazione e crea delle piccole variazione dell'onda che provocano tali rumori.

Bibliografia:

http://www.slideshare.net/valicac/visual-cryptography

http://www.dia.unisa.it/~ads/corso-security/www/CORSO-9900/vcs/index.htm

http://en.wikipedia.org/wiki/Visual_cryptography

http://users.telenet.be/d.rijmenants/en/visualcrypto.htm

http://homepage3.nifty.com/furumizo/gmaskd_e.htm