UNIVERSITÀ DEGLI STUDI DI SALERNO

Laurea Magistrale in Informatica

# Encryption Before Compression Coding Scheme for JPEG Image Compression Standard

*Progetto di Compressione dei Dati*


Prof.

Bruno Carpentieri

Partecipanti

Luigi Di Palma 0522501091

Domenico Nappi 0522500924

Toni Pannese 0522501047

Vincenzo Passariello 0522500973

Federica Ungherese 0522500942

# 1. Abstract

In questo articolo presentiamo un nuovo schema di compressione di immagini congiunto ad una fase di crittografia che potrà essere utilizzato assieme al popolare standard di compressione JPEG.
La crittografia viene eseguita prima della fase di compressione tramite trasformazioni lineari veloci, parametrizzate con una chiave privata, che non alterano le caratteristiche statistiche dei dati dell'immagine, il che permette all'algoritmo JPEG di mantenere le sue piene capacità di compressione.
L'articolo include anche un modello matematico dello schema proposto che permette un'analisi teorica dell'impatto della fase crittografica dell'immagine sul processo di compressione.
I risultati sperimentali ottenuti indicano che le qualità delle immagini ricostruite sono paragonabili a quelle ottenute tramite l'utilizzo dello standard JPEG senza la fase crittografica, inoltre, variando la qualità dell'immagine è possibile ottenere un rapporto di compressione differente.
Inoltre, nella nostra implementazione dello schema, per ciascuna immagine di test è stata effettuata una prova d'attacco da parte di una terza persona che intercetta l'immagine cifrata sul canale di comunicazione e tenta di decifrarla utilizzando una chiave differente da quella utilizzata nel processo di cifratura, senza successo.

# 2. Introduzione

L'incrementale sviluppo delle tecnologie di telecomunicazione e il conseguente aumento della larghezza di banda di trasmissione hanno permesso la diffusione di sistemi remoti di scambio d'immagini che vengono utilizzati per effettuare, ad esempio, videoconferenze o pubblicazione di contenuti multimediali nei social network.
Tuttavia, la trasmissione di queste immagini avviene tramite canali aperti il ché le rende vulnerabili all'intercettazione e all'ascolto da parte di destinatari non autorizzati.
La soluzione a questo problema consiste nell'utilizzare algoritmi crittografici.

Nelle realizzazioni pratiche dei sistemi di crittografia di immagini possiamo distinguere due approcci:

1. L'approccio **Compress-Then-Encrypt (CTE),** in cui la crittografia viene applicata dopo la compressione oppure avvengono in maniera simultanea.
2. L'approccio **Encrypt-Then-Compress (ETC),** dove i dati vengono in primo luogo criptati e successivamente sottoposti a compressione.

Appare subito chiaro che l'approccio ETC sia più difficile da realizzare in quanto lo scopo della crittografia è quello di nascondere tutte le correlazioni presenti nell'immagine mentre

il processo di compressione permette di ridurre la dimensione dei dati in base alla loro similarità e la risultante ridondanza di rappresentazione.

## *2a. Scenario codifica Encrypt-Then-Compress*

Gli scenari tipici di codifica ETC coinvolgono due attori, Alice e Bob, che hanno esigenze differenti e fiducia limitata tra loro.
Supponiamo che Alice voglia inviare delle immagini con contenuto confidenziale sulla rete o salvarli in un archivio esterno.
Bob è il responsabile della realizzazione di entrambi i compiti.
Tuttavia Alice non può fidarsi di Bob o dei dati che gli vengono inviati su un canale aperto.
Questo è il motivo per cui Alice intraprende il processo di crittografia dei dati, poiché è interessata alla confidenzialità di questi. D'altra parte, Bob è interessato alla compressione dei dati, e cioè alla riduzione della loro dimensione, poiché questo abbasserebbe i requisiti per lo spazio disponibile sul disco o la larghezza di banda necessaria dei canali di trasmissione dei dati.
Nel caso ottimale, Alice invia solo il testo cifrato a Bob che a sua volta, senza alcuna informazione aggiuntiva, è in grado di comprimere con l'uso di standard e strumenti noti.

## *2b. Stato dell'arte*

Gli schemi ETC proposti in letteratura si basano su vari approcci e permettono di ottenere valori differenti del **rapporto di compressione (CR)** e del **Peak Signal-to-Noise Ratio (PSNR)** che rappresenta una misura che viene adottata per valutare la qualità di un'immagine compressa rispetto all'originale, solitamente espresso in decibel (dB).

Gli approcci esistenti si basano su:

- Codifica della sorgente con informazioni aggiuntive (CR = 3, senza perdite)
- Ricostruzione iterativa dell'immagine basata sulla rappresentazione ridotta nel dominio della trasformazione ortogonale (CR = 3, PSNR = 35 dB)
- **Crittografia e compressione nel dominio della trasformazione Wavelet intera (CR = 30, PSNR = 30 dB)**
- Campionamento compressivo (CR = 2, PSNR = 30 dB)
- Elementi della teoria dei giochi (CR = 2.32, PSNR = 21 dB)

La migliore delle soluzioni esistenti, **Crittografia e compressione nel dominio della trasformazione Wavelet intera**, permette di ottenere risultati paragonabili a quelli caratteristici dello schema di compressione JPEG, tuttavia, non è compatibile con gli strumenti esistenti e il suo utilizzo richiederebbe nuovi standard di compressione delle immagini.

Per questo motivo è stato proposto un nuovo schema per la crittografia e la compressione di immagini secondo lo scenario ETC.

# 3. Schema proposto

La fase di crittografia sfrutta le trasformazioni lineari ortogonali e l'approccio di miscelazione dei dati dell'immagine all'interno di blocchi selezionati da aree differenti dell'immagine. Tale operazione permette di mantenere le caratteristiche statistiche dell'immagine (entropia), cruciale dal punto di vista della fase successiva di compressione.
La fase di compressione adotta un metodo classico basato sulla quantizzazione scalare nel dominio della trasformazione lineare, in pratica la **Discrete Cosine Transform (DCT)** e la codifica dell'entropia.
L'implementazione pratica di questo metodo è lo **standard JPEG per la compressione lossy di immagini**.

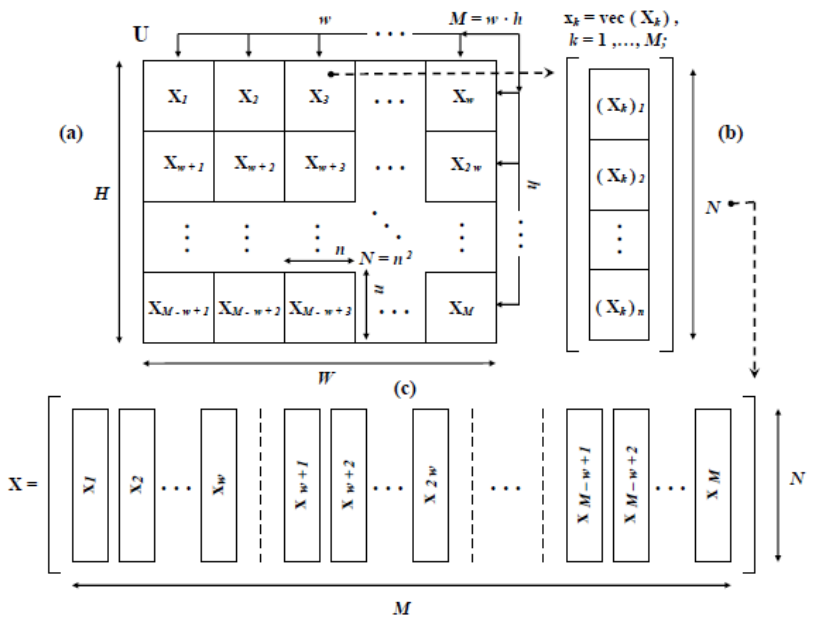
Quindi, lo schema proposto permette la realizzazione pratica dello scenario ETC con l'uso di strumenti di compressione standard, mantenendo l'efficienza della compressione a un livello paragonabile ad un caso senza la fase di crittografia.

## *3.a Data preparation*

Prima di subentrare nelle fasi principali dello schema Encrypt-Then-Compress occorre svolgere una fase di preprocessing. Per fare ciò, occorre prima definire l'input considerato.

Assumiamo che l'input sia un'immagine monocromatica. Siano $w$, $h$, $n \in \mathrm{N}$ e $\mathrm{W} = w \times n$, $\mathrm{H} = h \times n$ e siano $\mathrm{M} = w \times h$ e $\mathrm{N} = n^2$. L'immagine è rappresentata da una **matrice U** di H×W elementi denotati come $u_{ij} \in \mathrm{R}$, per $i = 1,\ldots,\mathrm{H}$ e $j = 1,\ldots,\mathrm{W}$.

Al fine di adattare l'input allo schema, si procede, prima di effettuare la cifratura e la compressione, a un **data preparation stage** che si articola in tre step, illustrati schematicamente in figura e descritti di seguito.




**Step 1**

L'immagine di input U è suddivisa in **frammenti quadrati $\mathbf{X_k}$**, $k$ = 1,…,M, di dimensione $n \times n$, ciascuno dei quali rappresenta uno degli elementi reali della matrice.

Le componenti della matrice U rappresentativa dell'immagine vengono mappate sugli elementi delle sottomatrici $X_k$ mediante la seguente relazione.

$\forall k \in 1,\ldots,M$ e $\forall i, j \in 1,\ldots,n$ si ha:

$$(\mathbf{X}_k)_{ij} = u_{\lfloor (k-1)/w \rfloor \times n + i,\ ((k-1)\ mod\ w) \times n + j}$$

**Step 2**

Le matrici $X_k$ vengono trasformate, mediante l'operatore di vettorizzazione *vec*, in **vettori colonna $\mathbf{x_k}$** di N = $n^2$ elementi in questo modo:

$$\mathbf{x}_k = vec\,(\mathbf{X}_k) = [\ (\mathbf{X}_k)_1^T\ \ (\mathbf{X}_k)_2^T\ \ \ldots\ \ (\mathbf{X}_k)_n^T\ ]^T,$$

Nella formula, $(X_k)_l$ , con $k$ = 1,…,M, $l$ = 1,…,$n$, è la l-esima colonna della matrice $X_k$.

**Step 3**

I vettori $x_k$ ottenuti allo step 2 vengono disposti in maniera consecutiva di modo da formare le colonne di una **matrice X** di N×M elementi.

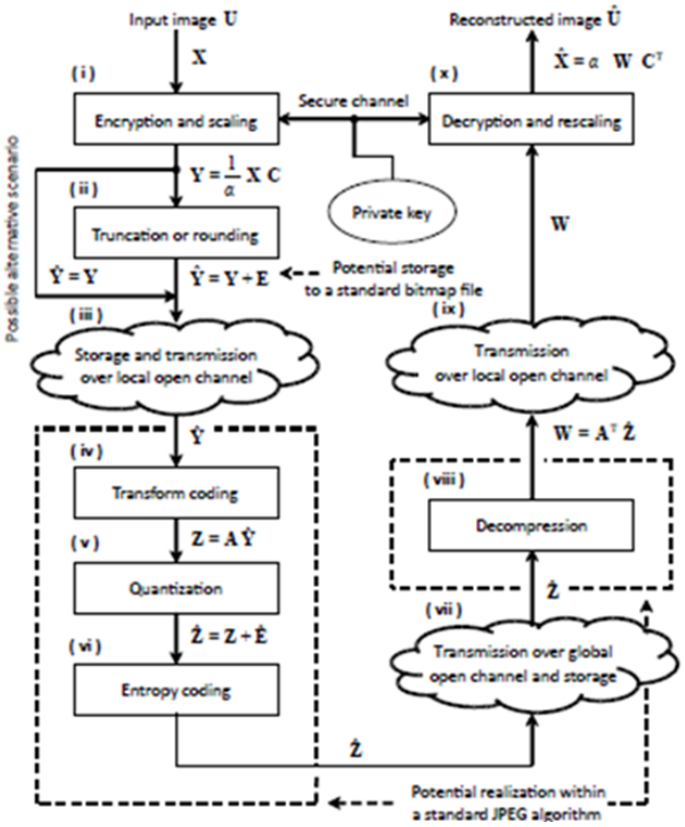
$$\forall k \in \{1, \ldots, M\} \quad (\mathbf{X})_k = \mathbf{x}_k$$

Nella formula, $(X)_k$ è la k-esima colonna della matrice X; quest'ultima sarà l'input definitivo dello schema considerato.

Dopo la preparazione della matrice X, può cominciare il processo di cifratura.

## *3.b Modello matematico*

La figura mostra chiaramente la sequenza di fasi che andranno a comporre e a caratterizzare la soluzione proposta nell'articolo.




Per grandi linee, procedendo con ordine, troviamo come l'input (che è la nostra immagine U di test resa in forma matriciale dalla procedura di data preparation, descritta dalla mia

collega precedentemente) entra in una prima fase di **Encryption and Scaling**. Dal punto di vista matematico, l'output di questo primo step sarà una nuova matrice **Y = X C**.

Il parametro α rappresenterà un fattore di scala, di solito maggiore di uno, utile per poter rendere accettabile, da un punto di vista matematico, il crittogramma **XC** nel corso dei vari steps. La matrice di crittografia **C,** che illustreremo nel corso della nostra presentazione, sarà una matrice ortogonale. Ricorderemo che in algebra lineare una matrice sarà denotata come ortogonale solo se invertibile e se la sua trasposta coincide con la sua inversa, in altre parole se $\mathbf{CC^T = I}$**,** dove **I** è la matrice identità.

Il secondo step della soluzione proposta sarà facoltativo, si adopererà una sorta di troncamento o arrotondamento, utile per alcuni formati standard come bitmap, il quale tratterà le immagini solo con valori di tipo intero.

La fase iv) prenderà come input una matrice che sarà o uguale all'espressione **X C** oppure all'espressione **Y + E** (solo se risulta essere avvenuto il troncamento e quindi la fase **ii)** ).

La fase iv), v), vi) sono rappresentate come un unico blocco perché rappresentative, anche se in maniera piuttosto esemplificativa, dell'algoritmo JPEG nella sua modalità baseline. In figura troviamo la presenza del blocco relativo al Transform Coding, dal momento che lo standard JPEG si serve di una Forward DCT per traslare il problema dal dominio spaziale ad un dominio frequenziale. Il blocco relativo alla fase di Quantizzazione è deputato alla fase di filtraggio mediante l'uso di matrici di quantizzazione, dettate stesso dallo standard o calibrate dall'utente. Lo step vi) è caratterizzato dall'ultima fase dello standard JPEG di codifica entropica, mediante Huffman o Aritmetic Coding.

Lo schema procede con il ramo inverso dove le protagoniste sono la fase di decompressione, invio sul canale pubblico e decodifica dell'immagine per mezzo della chiave privata.

L'intero modello potrebbe essere descritto, da un punto di vista matematico, mediante la seguente equazione: $\hat{X} = \alpha A^T ( A ( \alpha^{-1} X C + E ) + \hat{E} ) C^T$.

Questa formula altro non è che la composizione matriciale dell'immagine in arrivo al receiver.

Nell'ultima formula è stata esplicitamente indicata la matrice **W,** ottenuta dopo la fase viii) di decompressione.

Nell'articolo verrà calcolato, come primo valore, l'errore quadratico medio **D,** indicativo della bontà della ricostruzione. Da un punto di vista matematico, l'errore quadratico medio altro non è che un indice di dispersione statistico che servirà a capire di quanto l'output si potrà discostare dall'input. Questo oggetto matematico ci servirà in seguito per calcolare

l'errore quadratico medio in dipendenza dal bitrate R, per partorire così il tasso di distorsione e valutare quanto «spingerci» a comprimere per mantenere l'errore entro determinati limiti.

Una fase iniziale:

$$D = \frac{1}{MN}\, tr\{(\hat{\mathbf{X}} - \mathbf{X})(\hat{\mathbf{X}} - \mathbf{X})^T\}$$

dove tr{+} rappresenta la traccia della matrice

Prendendo in considerazione le stime delle **matrici di autocovarianza**, rispettivamente dell'errore di proiezione e dell'errore di quantizzazione:

$$\mathbf{R}_e \equiv 1/M\, \mathbf{E}\mathbf{E}^T \quad \text{e} \quad \mathbf{R}_{\hat{e}} \equiv 1/M\, \hat{\mathbf{E}}\hat{\mathbf{E}}^T$$

e, concentrandoci sugli elementi delle rispettive diagonali principali, indicative della misura di varianza di tali errori:

$$\sigma^2_{e_i} \equiv (\mathbf{R}_e)_{ii} \quad \text{e} \quad \sigma^2_{\hat{e}_i} \equiv (\mathbf{R}_{\hat{e}})_{ii}$$

sostituendo, l'espressione dello scarto quadratico medio sarà pari a:

$$\begin{aligned} D &= \alpha^2 \frac{1}{MN}\, tr\{((\mathbf{A}^T\hat{\mathbf{E}} + \mathbf{E})\mathbf{C}^T)(\mathbf{C}(\hat{\mathbf{E}}^T\mathbf{A} + \mathbf{E}^T))\} \\ &= \alpha^2 \frac{1}{MN}(tr\{\mathbf{E}\mathbf{E}^T\} + tr\{\hat{\mathbf{E}}\hat{\mathbf{E}}^T\}) = \alpha^2 \frac{1}{N}\left(\sum_{i=1}^N \sigma^2_{e_i} + \sum_{i=1}^N \sigma^2_{\hat{e}_i}\right). \end{aligned}$$

I termini relativi alle varianze possono essere approssimati secondo gli studi della teoria di quantizzazione, nel seguente modo:

I valori di varianza potranno essere approssimati secondo gli studi della teoria di quantizzazione nel seguente modo:

$$\sigma^2_{\hat{e}_i} \cong \frac{\Delta_i^2}{12}, \quad i = 1, \ldots, N, \quad \text{e} \quad \sigma^2_{e_i} \cong \frac{1}{3}, \quad i = 1, \ldots, N.$$

In modo tale che **l'errore quadratico medio**, sostituendo, diventerà:

$$D \cong \alpha^2 \frac{1}{N} \sum_{i=1}^N \frac{\Delta_i^2}{12} + \alpha^2 \sigma_e^2$$

Una volta calcolata l'espressione relativa all'errore quadratico medio, potremmo concentrarci sul calcolo della media del minimo bit rate R, rappresentativo del bit rate così come lo intende la Teoria di Distorsione di Shannon.
R sarà il bit rate rappresentativo del singolo campione codificato nell'output dello schema proposto in una slide precedente, D sarà l'errore quadratico medio relativo al singolo campione di output.
Partendo da una forma esplicita della matrice-crittogramma Z, calcolando anche qui la stima della sua matrice di autocovarianza, prendendo sempre in considerazione gli elementi della diagonale principale (indicativi dei valori di varianza), otteniamo le espressioni indicate nella slide seguente (iteriamo il procedimento anche con la matrice **X**):

$$\mathbf{Z} = \mathbf{A}\,(\,\alpha^{-1}\,\mathbf{X}\,\mathbf{C} + \mathbf{E}\,)$$

$$\mathbf{R}_z \equiv 1/M\;\mathbf{Z}\,\mathbf{Z}^T \quad \text{con} \quad \sigma^2_{z_i} \equiv (\,\mathbf{R}_z)_{ii}$$

$$\mathbf{R}_x \equiv 1/M\,\mathbf{X}\mathbf{X}^T \quad \text{con} \quad \sigma^2_{x_i} \equiv (\mathbf{R}_x)_{ii}$$

Abbiamo che, esplicitando:

$$\mathbf{R}_z = \frac{1}{M}\,(\,\mathbf{A}\,(\,\alpha^{-1}\,\mathbf{X}\,\mathbf{C} + \mathbf{E}\,)\,)\,(\,\mathbf{A}\,(\,\alpha^{-1}\,\mathbf{X}\,\mathbf{C} + \mathbf{E}\,)\,)^T = \alpha^{-2}\,\mathbf{A}\,\mathbf{R}_x\mathbf{A}^T + \mathbf{A}\,\mathbf{R}_e\mathbf{A}^T.$$

con

$$\sigma^2_{z_i} = \alpha^{-2}\,\mathbf{a}_i^T\,\mathbf{R}_x\,\mathbf{a}_i + \sigma^2_e,\;\; i = 1, \ldots, N$$

In accordo con il **Teorema del Limite Centrale**, assunto che il valore di M è molto grande, possiamo di certo concludere che i valori della matrice **Z** seguono una distribuzione normale di questo tipo:

$$\mathbf{z}_i^T \simeq \mathcal{N}(\,0,\,\sigma^2_{z_i}\,)\;,\; i = 1, \ldots, N.$$

Prendendo in prestito gli studi relativi al **rate di distorsione di Shannon** sappiamo che l'entropia di una qualsiasi variabile X, quantizzata con gli step $\Delta \in \mathbb{R}^+$, potrà essere approssimata nel seguente modo:

$h\,(X) - log_2\,\Delta$ dove **h (X) è il differenziale di entropia** della variabile X.

Nel nostro caso, questo differenziale di entropia sarà pari a $\frac{1}{2}log_2(2\pi e\sigma^2_x)$ perché segue una distribuzione normale.

La dimostrazione dell'indipendenza della matrice di crittografia C è dimostrata dal fatto che essa non compare nella formulazione della D(R). L'indipendenza ci garantisce che il lavoro iniziale crittografico non influirà, in termini prestazionali, sul lavoro di compressione. La dimostrazione che riusciamo ad ottenere con la formulazione D(R) verrà indicata nel seguito:

In ultimo abbiamo che l'espressione per il **bitrate** sarà pari a:

$$R \cong \frac{1}{N}\sum_{i=1}^{N}\left(\tfrac{1}{2}\,log_2(2\pi e\sigma_{z_i}^2) - log_2\Delta_i\right) = \frac{1}{N}\sum_{i=1}^{N}\tfrac{1}{2}\,log_2\left(\frac{2\pi e}{\alpha^2}\cdot\frac{\mathbf{a}_i^T\,\mathbf{R}_x\,\mathbf{a}_i\,+\,\alpha^2\sigma_e^2}{\Delta_i^2}\right)$$

$$= \tfrac{1}{2}\,log_2\left(\frac{2\pi e}{\alpha^2}\left(\prod_{i=1}^{N}\frac{\mathbf{a}_i^T\,\mathbf{R}_x\,\mathbf{a}_i\,+\,\alpha^2\sigma_e^2}{\Delta_i^2}\right)^{\frac{1}{N}}\right).$$

Introducendo il fattore:

$$\beta^2 = \frac{2\pi e}{\alpha^2}\left(\prod_{i=1}^{N}\frac{\mathbf{a}_i^T\,\mathbf{R}_x\,\mathbf{a}_i\,+\,\alpha^2\sigma_e^2}{\Delta_i^2}\right)^{\frac{1}{N}}\cdot\,2^{-2R}.$$

Otteniamo, procedendo per varie sostituzioni, alla formulazione dell'errore quadratico medio in dipendenza del bitrate, tale formulazione prenderà il nome di **Funzione di rate di distorsione**.

$$D(R)\,\cong\,\frac{\pi e\,\|\mathbf{\Delta}\|^2}{6N}\left(\prod_{i=1}^{N}\frac{\mathbf{a}_i^T\,\mathbf{R}_x\,\mathbf{a}_i\,+\,\alpha^2\sigma_e^2}{\Delta_i^2}\right)^{\frac{1}{N}}\cdot\,2^{-2R}\,+\,\alpha^2\sigma_e^2\,,$$

# 4. Risultati dello schema proposto

L'analisi dell'efficienza della crittografia è un passo essenziale nella progettazione di un qualsiasi sistema crittografico. Tale analisi può essere effettuata:

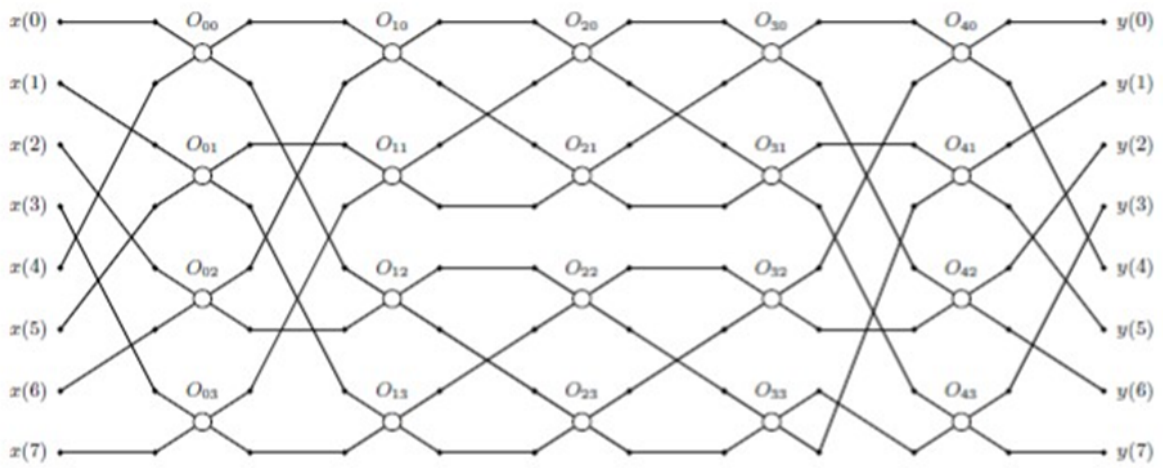
- Sulla base delle misure note utilizzate per valutare l'efficacia dei metodi crittografici di immagine, cioè, analisi combinatoria e indicatori di qualità sotto forma di Histogram Analysis (HA), Deviazione massima (MD), coefficiente di correlazione (CC) o deviazione irregolare (ID);
- Come un'analisi statistica volta a ricavare dipendenze che permettono di determinare la probabilità di ottenere un errore di ricostruzione ad un livello non superiore ad un dato valore nel caso di un tentativo di indovinare causalmente la chiave di cifratura corretta.

Al fine di verificare praticamente l'efficienza dei processi di cifratura e compressione all'interno dello schema proposto, sono state effettuate diverse prove su immagini naturali con caratteristiche statistiche differenti, andando ad utilizzare, quindi, un approccio basato sull'analisi statistica.

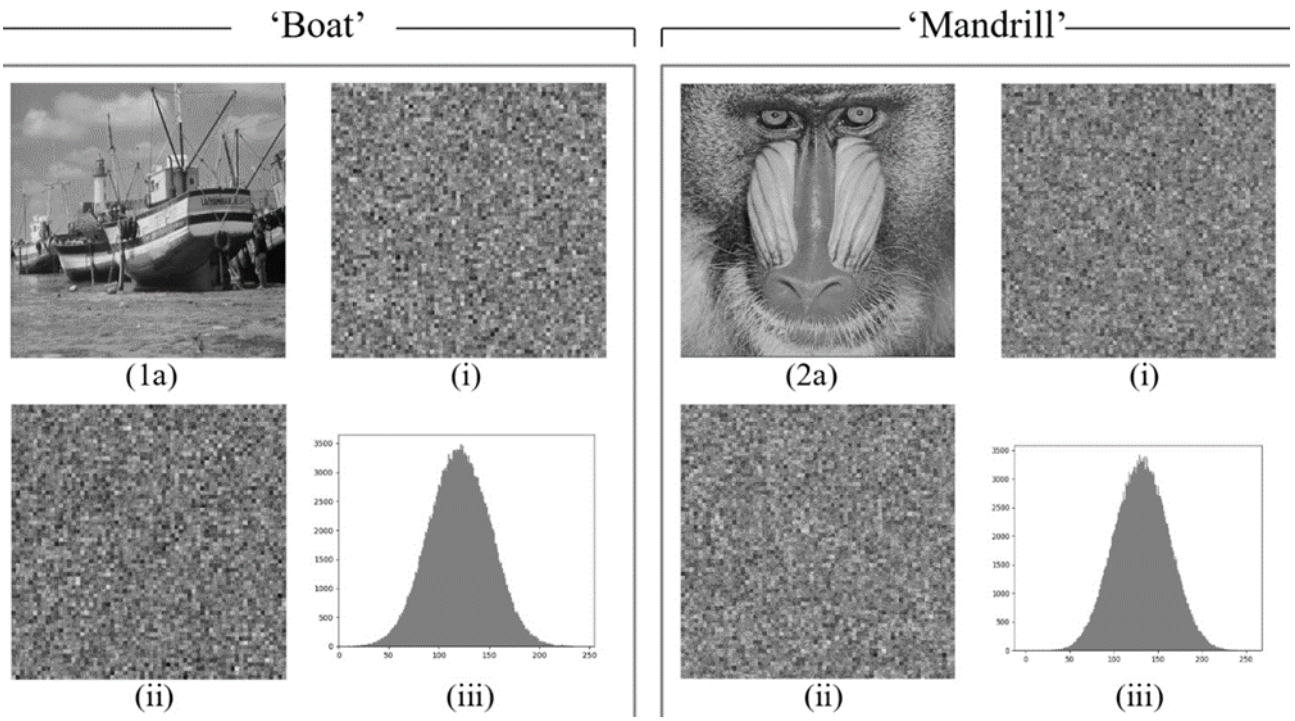
Sono state utilizzate quattro immagini di test standard: due hanno caratteristiche simili e sono caratterizzate da relativa fluidità; le altre, invece, contengono componenti ad alta frequenza. Tutte le immagini erano in scala di grigi a 8 bit con risoluzione 512x512 pixel ciascuna.

## *4.a Efficienza processo crittografico*

Per quanto riguarda il processo di cifratura dello schema proposto, è stata utilizzata una matrice di codifica ortogonale a blocchi, con dimensione del blocco K=8, implementata con una struttura parametrica veloce il cui diagramma di flusso dei dati è presentato nella figura sottostante:




Nella figura successiva, invece, sono mostrati due esempi di immagini con i loro moduli ottenuti dopo la cifratura (i) e dono la decifratura (ii) con chiave privata non valida. L'esame visivo dei risultati ottenuti indica la mancanza di somiglianza tra l'input e le immagini criptate. Inoltre, le distribuzioni di probabilità delle luminanze dei pixel nelle immagini ottenute dopo la cifratura hanno caratteristiche di rumore gaussiano (grafico (iii)).




## *4.b Efficienza processo di compressione*

In seguito, è stata analizzata l'efficienza relativa al processo di compressione.

Questa è la tabella che mostra il confronto dei risultati di compressione tra lo standard JPEG (STD) e l'approccio proposto all'interno di questo elaborato (ETC) per tutte le immagini di prova utilizzate.

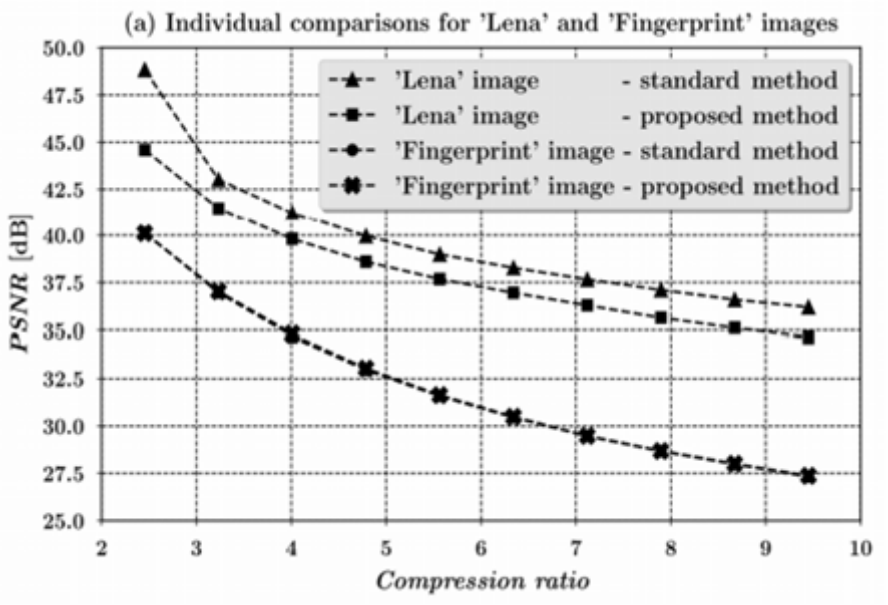
| Comp. ratios | PSNR values [dB] | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 'Lena' | | 'Fingerprint' | | 'Boat' | | 'Mandrill' | | Average values | |
| | STD | ETC | STD | ETC | STD | ETC | STD | ETC | STD | ETC |
| 2.46 | 48.84 | 44.59 | 40.13 | 40.11 | 42.54 | 42.42 | 36.25 | 36.03 | 41.94 | 40.79 |
| 3.23 | 43.02 | 41.49 | 36.96 | 37.03 | 39.59 | 39.00 | 32.57 | 32.24 | 38.03 | 37.44 |
| 4.01 | 41.24 | 39.81 | 34.69 | 34.81 | 37.76 | 37.06 | 30.31 | 29.94 | 36.00 | 35.41 |
| 4.79 | 39.97 | 38.60 | 32.96 | 33.04 | 36.55 | 35.68 | 28.84 | 28.36 | 34.58 | 33.92 |
| 5.56 | 38.99 | 37.68 | 31.59 | 31.64 | 35.61 | 34.56 | 27.74 | 27.26 | 33.48 | 32.78 |
| 6.34 | 38.27 | 36.95 | 30.46 | 30.50 | 34.82 | 33.66 | 26.92 | 26.45 | 32.62 | 31.89 |
| 7.12 | 37.66 | 36.29 | 29.48 | 29.48 | 34.19 | 32.86 | 26.24 | 25.82 | 31.89 | 31.11 |
| 7.90 | 37.10 | 35.64 | 28.68 | 28.69 | 33.65 | 32.25 | 25.72 | 25.32 | 31.29 | 30.47 |
| 8.67 | 36.59 | 35.12 | 27.98 | 28.01 | 33.15 | 31.67 | 25.28 | 24.90 | 30.75 | 29.93 |
| 9.45 | 36.20 | 34.64 | 27.32 | 27.38 | 32.66 | 31.13 | 24.89 | 24.53 | 30.27 | 29.42 |

È subito possibile notare che al variare del compression rate la differenza di qualità tra i metodi, JPEG e ETC non supera i 4,25 dB per l'immagine di Lena, mentre per l'immagine dell'impronta digitale lo schema proposto presenta prestazioni addirittura più efficienti dello standard stesso.

Questo effetto può essere dovuto dalla codifica **Run-lenght encoding (RLE)** dei coefficienti della trasformata discreta del coseno (DCT) che viene usata nell'ultima parte dell'algoritmo JPEG. Sebbene lo schema di cifratura dell'immagine tende a preservare le matrici di autocovaranzia globale, non preserva le caratteristiche dei pixel locali, il che, nel caso di immagini statistiche non uniformi ( ovvero le immagini con regioni sia di pixel omogenei che eterogenei, come Lena) peggiora l'efficienza del passo RLE del JPEG all'interno dello schema proposto, accorciando la lunghezza dei runs (sequenza in cui un dato viene ripetuto più volte consecutivamente) degli zeri dei coefficienti DCT quantizzati, rispetto alla lunghezza ottenuta dal metodo senza cifratura.

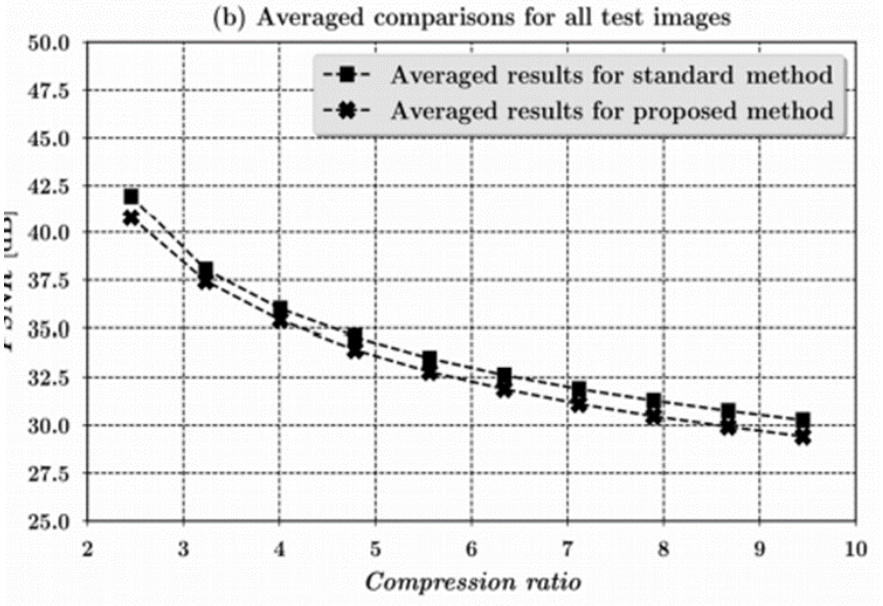
Per quanto riguarda invece le immagini che presentano caratteristiche statistiche uniformi, come l'impronta digitale, il passo RLE produce lunghezze simili degli zeri dei coefficienti DCT per entrambi gli approcci (JPEG e ETC), quindi, non influenza il rapporto complessivo ottenuto nello schema proposto.

Dal seguente grafico è possibile vedere ancora meglio il risultato degli algoritmi a confronto e come la compressione dell'immagine di Lena da parte dello standard JPEG risulti, anche se di poco, più performante della soluzione proposta da questo elaborato (ETC).




Mentre per l'immagine dell'impronta digitale i risultati sono pressoché identici se non migliori da parte dello schema ETC.

Nel seguente grafico sono riportati i risultati medi dei due approcci, è possibile vedere che il processo di compressione della soluzione proposta risulta essere, anche se di poco, meno ottimale dello standard JPEG. Tuttavia, può essere considerato efficiente.



# 5. Realizzazione del nostro schema

## *5a. Scelta, progettazione della matrice di crittografia C e preprocessing dell'immagine*

Seguendo la linea progettuale del paper assegnatoci, il cuore del processo che coinvolge la crittografia dell'immagine risiede nella scelta della matrice C di crittografia.

Come operazione preliminare si è scelto di trasformare le immagini, sempre seguendo l'idea del paper, in forma matriciale, utilizzando la libreria numpy.

L'idea del team è stata quella di ottenere, dall'immagine di partenza, le tre componenti Y, U e V, rappresentative della matrice di luminanza (più sensibile all'occhio umano) e delle due matrici di crominanza (meno sensibili all'occhio umano). La scelta di utilizzare la matrice Y è stata partorita dal fatto che il paper assegnatoci menzionava solo immagini in scala di grigio, la stessa tipologia di immagini su cui il nostro team ha incentrato il suo lavoro. Nella sezione "Conclusioni e Sviluppi Futuri" si proverà a dare un'idea implementativa per il lavoro di immagini a colori.

Nel nostro progetto le matrici di immagini sono state elaborate per mezzo della libreria NumPy. Quest'ultima è una libreria open source per il linguaggio di programmazione Python, che aggiunge supporto a grandi matrici e array multidimensionali insieme a una vasta collezione di funzioni matematiche di alto livello per poter operare efficientemente su queste strutture dati.

Le immagini utilizzate sono state prese da un dataset pubblico sulla piattaforma Kaggle, reperibile al link[3]. Kaggle è una comunità online di data scientist e professionisti dell'apprendimento automatico . Esso consente agli utenti di trovare e pubblicare set di dati, esplorare e creare modelli in un ambiente di scienza dei dati basato sul Web, lavorare con altri data scientist e ingegneri dell'apprendimento automatico e partecipare a concorsi per risolvere le sfide della scienza dei dati.

Nel nostro personale metodo ETC è stata anche trasformata la matrice rappresentativa dell'immagine avendo cura di convertire le varie entry in tipi di dati intero tramite la funzione **astype()**, così come indicava il paper per il suo metodo opzionale (qualora si volesse, ad esempio, lavorare con standard specifici tipo il .bitmap, caratterizzato da immagini le cui matrici hanno solo valori di tipo intero.

Ritornando alla nostra matrice di crittografia, la scelta del team è virata su un tipo di matrice ortonormale. In matematica, e più precisamente in algebra lineare, una matrice ortonormale è una matrice invertibile la cui trasposta coincide con la sua inversa. Nel processo di decompressione è fondamentale poter riottenere la matrice inversa per

realizzare il processo di ricostruzione dell'immagine. Nel nostro caso specifico, ottenere una matrice trasposta equivale ad ottenere la sua inversa.

Dal momento che la matrice di crittografia assume il ruolo di chiave privata, si è scelto di generare i suoi valori in maniera del tutto randomizzata.

Il listato prodotto per il calcolo della matrice di crittografia verrà presentato di seguito, utilizzando, anche in questo caso, la libreria NumPy di Python.

```
# Funzione per il calcolo della matrice C ortogonale
def rvs(dim=512): # dim è la dimensione dell'immagine n x n
    random_state = np.random # elementi scelti in maniera casuale
    H = np.eye(dim)
    D = np.ones((dim,))
    for n in range(1, dim):
        x = random_state.normal(size=(dim-n+1,))
        D[n-1] = np.sign(x[0])
        x[0] -= D[n-1]*np.sqrt((x*x).sum())
        # Householder transformation
        Hx = (np.eye(dim-n+1) - 2.*np.outer(x, x)/(x*x).sum())
        mat = np.eye(dim)
        mat[n-1:, n-1:] = Hx
        H = np.dot(H, mat)
        # Sistema l'ultimo segno affinché il determinante sia = 1
    D[-1] = (-1)**(1-(dim % 2))*D.prod()
    # Prodotto equivalente a np.dot(np.diag(D), H)
    H = (D*H.T).T
    return H
```

Ottenere l'immagine criptata sarà facilmente ottenibile realizzando il prodotto matriciale tra la matrice dell'immagine A e la matrice di crittografia C, mediante la seguente istruzione:

```
# Cifratura
def encrypt(numpydata_image, name):
  # salvataggio matrice di cifratura come array numpy
  C = rvs()
  # Prodotto tra la matrice di cifratura e la matrice dell'immagine
  # L'operatore @ viene utilizzato per effettuare il prodotto tra matrici
  A = numpydata_image @ C
```

## *5b. Processo di compressione, decifratura e decompressione*

Una volta effettuato il processo di cifratura, si procede alla fase di compressione.

In Python, per poter effettuare il processo di compressione di un'immagine servendosi dello standard JPEG è possibile utilizzare la libreria PIL (*Python Image Library*), che costituisce una solida base per svolgere operazioni basilari di image processing.

La **compressione** avviene mediante il metodo save() della libreria PIL, che consente di salvare l'immagine al path specificato e con l'estensione desiderata, nel nostro caso, come indicato nel paper di partenza, utilizzando JPEG baseline.

```
compress_image.save(newpath_compress, optimize = True, quality = qlt)
```

Il metodo consente anche di indicare dei parametri aggiuntivi, che nel nostro caso sono il parametro di ottimizzazione e della **qualità dell'immagine**; per quest'ultimo valore, che può variare tra 0 a 95, abbiamo optato, per avere maggiori casi di test che generassero diversi compression rate, per i seguenti valori, in ordine crescente di qualità: 10, 30, 50, 70, 90.

Dopo aver cifrato e compresso l'immagine, occorre risalire all'input iniziale. Per farlo, si ripercorre lo schema a ritroso, effettuando prima la decompressione e poi la decifratura.

Nel metodo implementato per la **decompressione**, si accede al path nel quale è stato memorizzato il file relativo all'immagine cifrata e compressa, e, una volta ottenuto accesso all'immagine, si procede in maniera analoga al metodo di compressione, stavolta memorizzando l'immagine tramite save() in un formato differente dal JPEG.

Per procedere alla fase di **decifratura**, occorre avere la chiave di decifratura, ossia la matrice ortonormale C calcolata precedentemente col metodo rvs().

L'algoritmo di decifratura moltiplica, mediante l'operatore @ per matrici, l'inversa della chiave per l'immagine cifrata. Poiché tale operazione è svolta tra matrici, è seguita dalla conversione della matrice ottenuta in un'immagine RGB tramite il metodo fromarray() e memorizzata.

Di seguito si riporta lo snippet di codice della funzione decrypt.

```python
def decrypt(A, C, name):
  #calcolo della matrice C inversa, essendo la matrice ortogonale l'inversa coincide con la trasposta
  Inv = np.linalg.inv(C)

  #ritorno all'immagine iniziale
  originale = A @ Inv

  image_decoded = Image.fromarray(originale)
  image_decoded = image_decoded.convert('RGB')

  # Calcolo del path da assegnare all'immagine decifrata e decompressa
  newpath_decoded = path_image_decoded.replace("decoded_image.jpg","decoded_image_"+name)
  image_decoded.save(newpath_decoded)
```

## *5c. Prova dell'attaccante*

Al fine di indagare la sicurezza dello schema, in maniera analoga a quanto svolto nel paper di partenza abbiamo previsto uno scenario nel quale un attaccante eavesdropper carpisce un'immagine cifrata e genera una chiave in maniera casuale per decifrare l'immagine. Tale scenario, per completezza, è stato simulato su tutte le immagini usate nei test, e lo schema non è stato mai rotto.

Di seguito sono riportate le immagini date in output dallo schema implementato.

*Immagine originale*

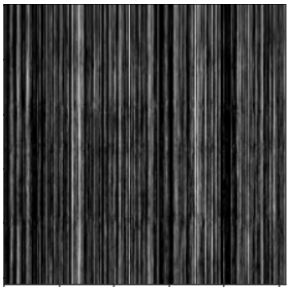


*Immagine cifrata*

*Immagine decifrata e decompressa*

Andando a generare una matrice ortonormale con lo stesso metodo rvs() usato in fase di cifratura, vista la natura randomica adottata nella generazione della chiave di crittografia, l'attaccante non riesce a risalirvi e a decifrare l'immagine che intercetta sul canale di comunicazione. Il risultato prodotto da questa decifratura erronea è un'immagine che presenta rumore, che è quasi indistinguibile dalla corrispettiva cifrata e che non è in alcun modo correlabile all'immagine in chiaro.

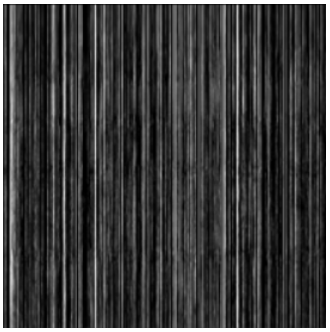


*Immagine decifrata e decompressa dall'attaccante usando una chiave casuale*

# 6. Risultati del nostro schema

## *6a. PSNR e Compression ratio*

Sulla base degli esperimenti che sono stati effettuati sono stati presi in considerazione dei particolari parametri che sono il **Peak Signal-to-Noise Ratio (PSNR)** quest'ultima risulta essere una misura utilizzata per valutare la qualità di un'immagine compressa rispetto a quella originale.

Il **PSNR** viene adoperato come misura di qualità all'interno della compressione di tipo lossy, maggiore sarà il valore del **PSNR** maggiore risulterà essere la "somiglianza" con l'immagine originale.

Di seguito verrà mostrata la formula per calcolarlo:

$$PSNR = 20 \cdot \log_{10}\left(\frac{MAX\{I\}}{\sqrt{MSE}}\right)$$

Dove **MAX{I}** rappresenta il valore massimo possibile di un pixel dell'immagine. Per le immagini in scala di grigio tale valore sarà 255, mentre per un'immagine binaria il valore sarà uguale ad uno.

**L'MSE** invece non è altro che l'errore quadratico medio che viene definito nel seguente modo:

$$MSE = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \|I(i,j) - K(i,j)\|^2$$

Indicheremo con I l'immagine originale mentre con K l'immagine compressa, entrambe di dimensione MxN.

Di seguito verrà riportato lo snippet di codice relativo al calcolo del PSNR.

```
def PSNR(encoded_path, encoded_compressed_path):
  original = cv2.imread(encoded_path) #immagine criptata
  compressed = cv2.imread(encoded_compressed_path, 1) #immagine criptata compressa

  mse = np.mean((original - compressed) ** 2)
  if(mse == 0): # MSE è uguale a 0 se non c'è rumore nell'immagine
    # In tal caso il PSNR non è rilevante
    return 100
  max_pixel = 255.0
  psnr = 20 * log10(max_pixel / sqrt(mse))
  return psnr
```

Il secondo parametro preso in considerazione è stato il compression ratio che rappresenta una misura della riduzione della dimensione dei dati prodotta da un algoritmo di compressione.

Il compression ratio è definito come il rapporto tra la dimensione dell'immagine non compressa e quella compressa.

$$\text{Compression Ratio} = \frac{\text{Uncompressed Size}}{\text{Compressed Size}}$$

Implementato nel seguente modo:

```
def cr(uncompressed_path, compressed_path):
  compression_rate = os.path.getsize(uncompressed_path) / os.path.getsize(compressed_path)
  print("Compressione Rate (size img iniziale/ size img compressa compressa): " + "{:.2f}".format(compression_rate))
  return compression_rate
```

Al fine di valutare differenti **compression ratio** per ciascuna delle immagini sono state prese in considerazione 5 differenti qualità.

## *6b. Risultati sperimentali*

Per verificare l'efficienza del nostro sistema, abbiamo effettuato dei test su un set di immagini prese per cinque qualità differenti. I test condotti coinvolgono un campione di 250 immagini, per ciascuna delle quali sono state considerate cinque qualità differenti per

un totale di 1250 casi di test; questi test sono stati parallelizzati tra i membri del gruppo per ottimizzare i tempi e rendere equo il lavoro di testing nel team di lavoro.

I risultati ottenuti sono stati salvati in singoli file ".csv", distinti in base alla qualità considerata. Di seguito sono riportati degli estratti del file contenente tali risultati:

| | Name | CR | PSNR-STD | PSNR-ETC |
|---|---|---|---|---|
| 0 | 3662865_qlt10.png | 9.67 | 32.38 | 30.32 |
| 1 | 4162702_qlt10.png | 12.52 | 29.3 | 29.95 |
| 2 | 5648321_qlt10.png | 7.56 | 32.93 | 30.55 |
| 3 | 5558592_qlt10.png | 11.9 | 33.4 | 30.05 |
| 4 | 5088155_qlt10.png | 12.35 | 30.32 | 29.75 |

Qualità 10

| | Name | CR | PSNR-STD | PSNR-ETC |
|---|---|---|---|---|
| 0 | 3662865_qlt30.png | 5.56 | 35.28 | 32.34 |
| 1 | 4162702_qlt30.png | 6.43 | 30.07 | 31.02 |
| 2 | 5648321_qlt30.png | 4.52 | 35.83 | 32.74 |
| 3 | 5558592_qlt30.png | 6.06 | 35.73 | 31.76 |
| 4 | 5088155_qlt30.png | 6.09 | 31.81 | 31.29 |

Qualità 30

| | Name | CR | PSNR-STD | PSNR-ETC |
|---|---|---|---|---|
| 0 | 3662865_qlt50.png | 4.4 | 36.82 | 33.72 |
| 1 | 4162702_qlt50.png | 4.65 | 30.73 | 31.93 |
| 2 | 5648321_qlt50.png | 3.59 | 37.44 | 34.67 |
| 3 | 5558592_qlt50.png | 4.63 | 36.9 | 33.02 |
| 4 | 5088155_qlt50.png | 4.55 | 32.88 | 32.21 |

Qualità 50

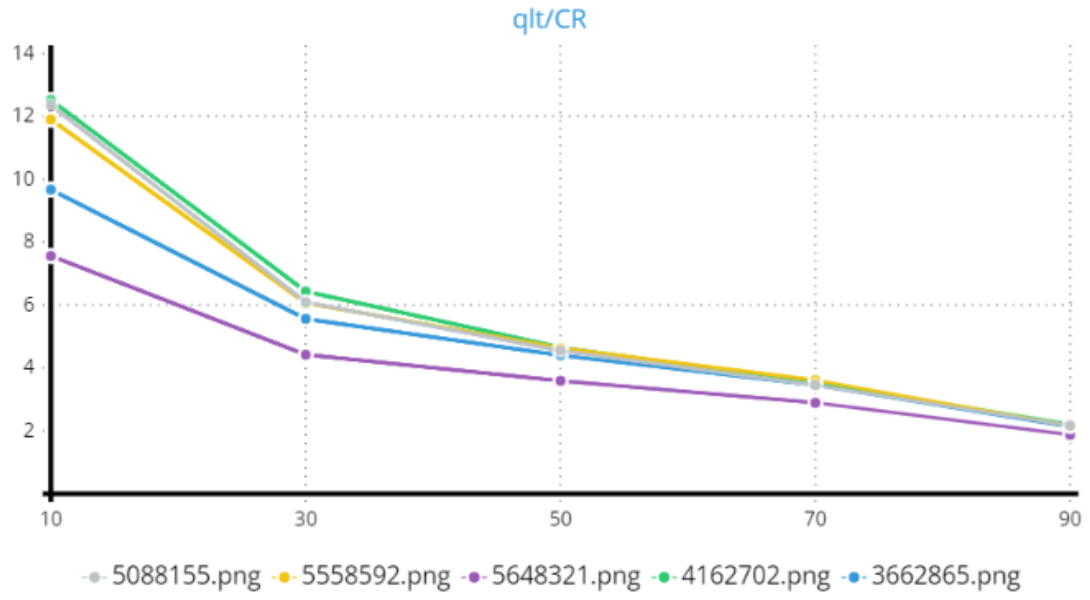
| | Name | CR | PSNR-STD | PSNR-ETC |
|---|---|---|---|---|
| 0 | 3662865_qlt70.png | 3.45 | 38.6 | 35.59 |
| 1 | 4162702_qlt70.png | 3.52 | 32.01 | 33.15 |
| 2 | 5648321_qlt70.png | 2.89 | 39.42 | 36.85 |
| 3 | 5558592_qlt70.png | 3.61 | 38.5 | 34.65 |
| 4 | 5088155_qlt70.png | 3.45 | 34.42 | 33.84 |

Qualità 70

| | Name | CR | PSNR-STD | PSNR-ETC |
|---|---|---|---|---|
| 0 | 3662865_qlt90.png | 2.13 | 43.63 | 40.58 |
| 1 | 4162702_qlt90.png | 2.21 | 38.06 | 38.82 |
| 2 | 5648321_qlt90.png | 1.87 | 44.85 | 43.27 |
| 3 | 5558592_qlt90.png | 2.16 | 44.01 | 40.79 |
| 4 | 5088155_qlt90.png | 2.17 | 40.22 | 39.61 |

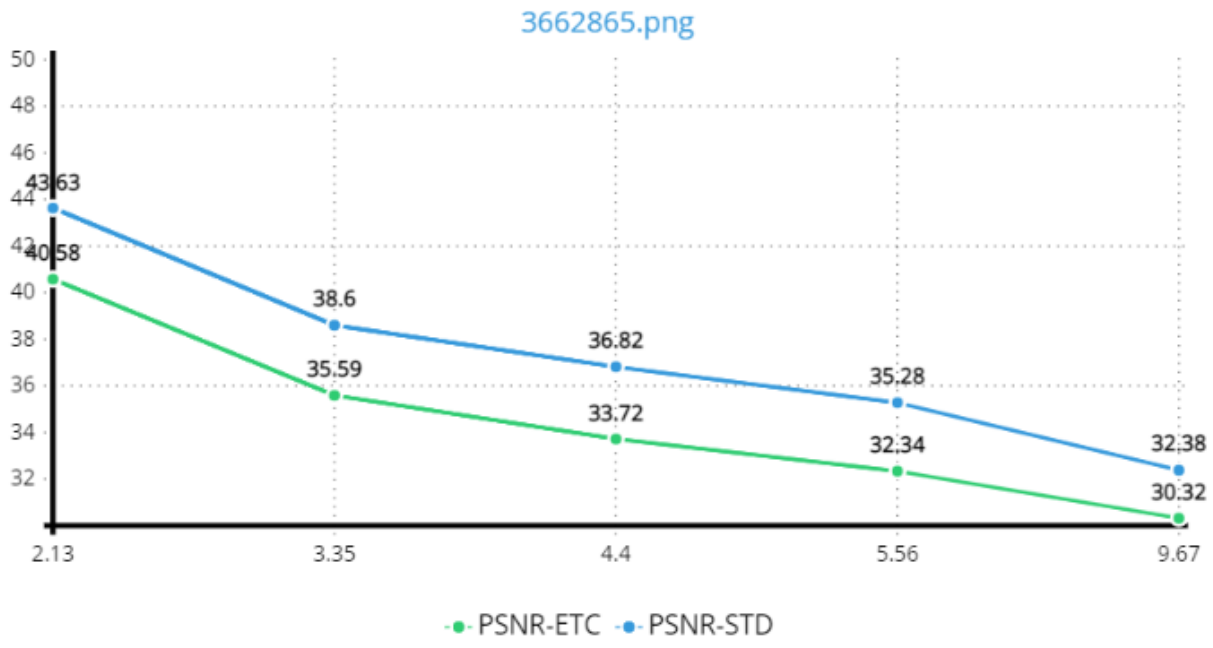
Qualità 90

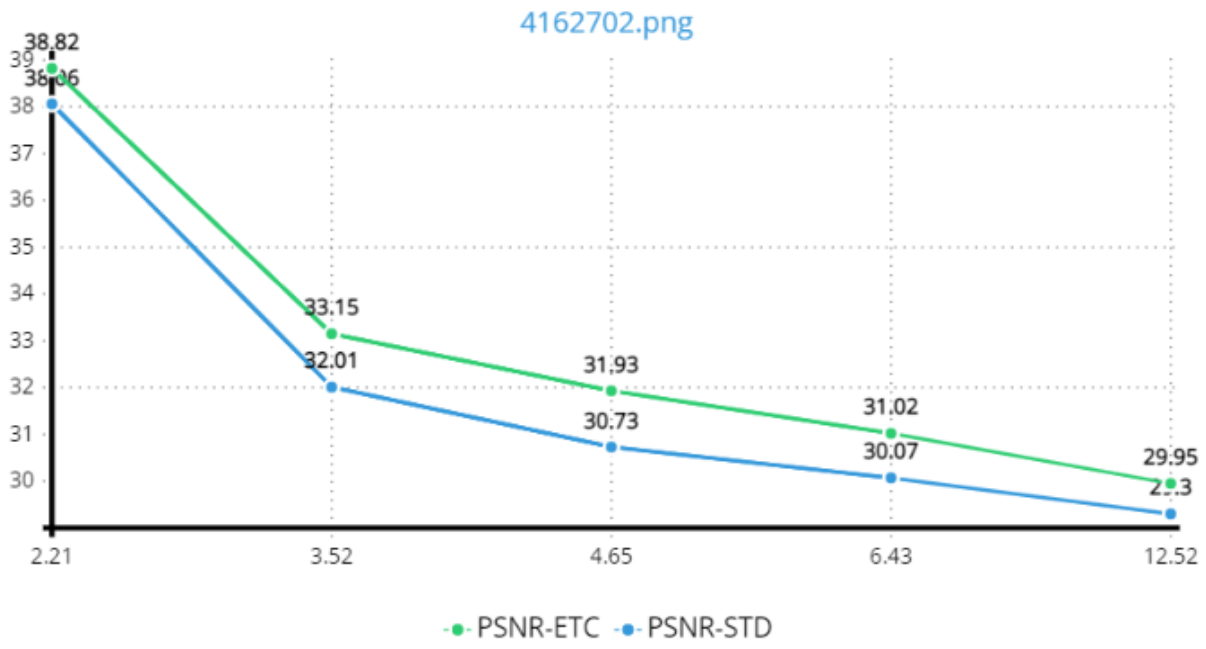
Andando ad analizzare i risultati ottenuti, è subito emerso come il Compression Rate cambi al variare della qualità delle immagini. In particolare, più è alta la qualità dell'immagine, minore è il Compression Rate.

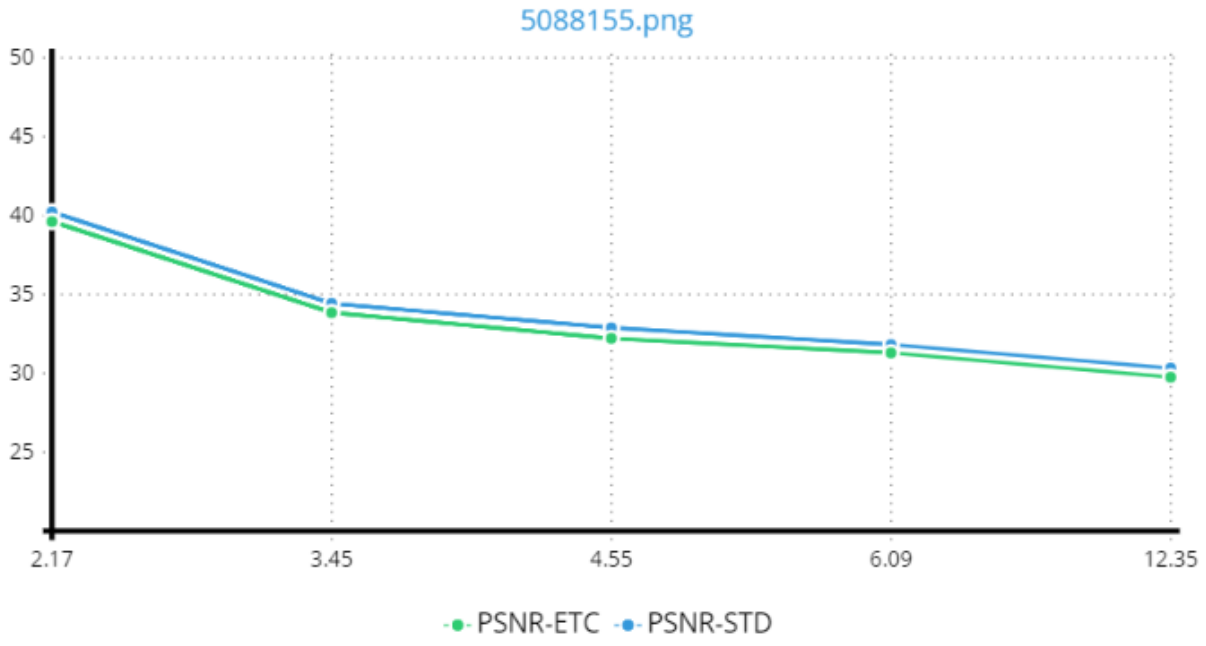


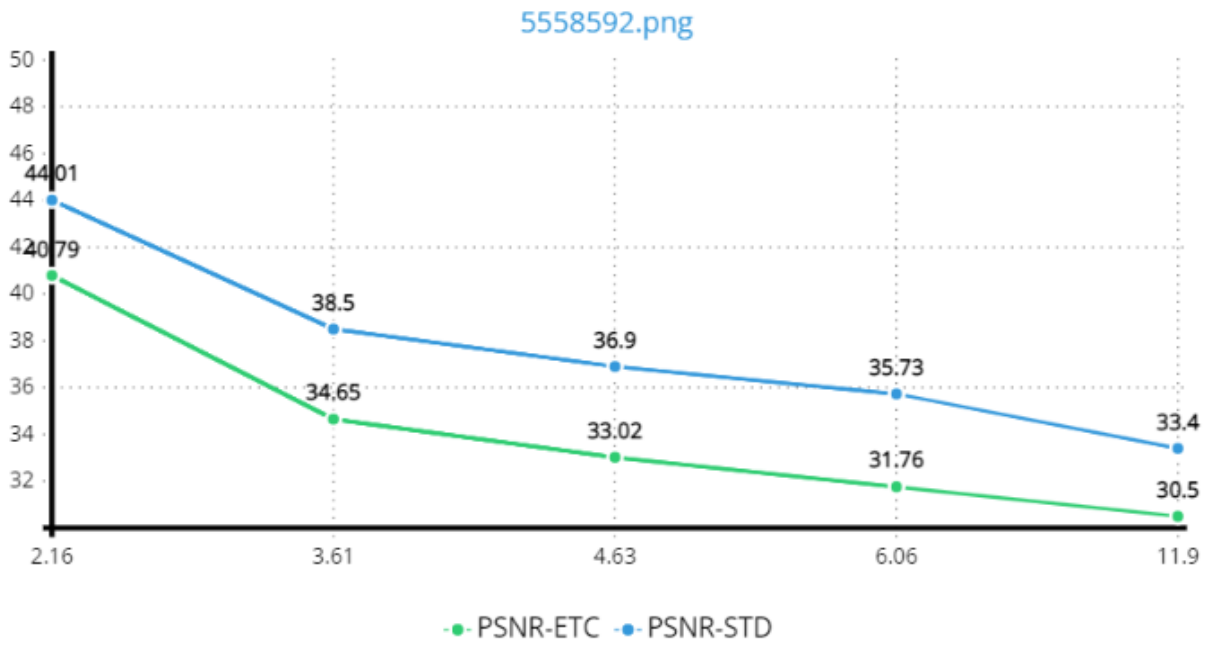

Variazione CR al variare della qualità d'immagine su un campione di 5 immagini

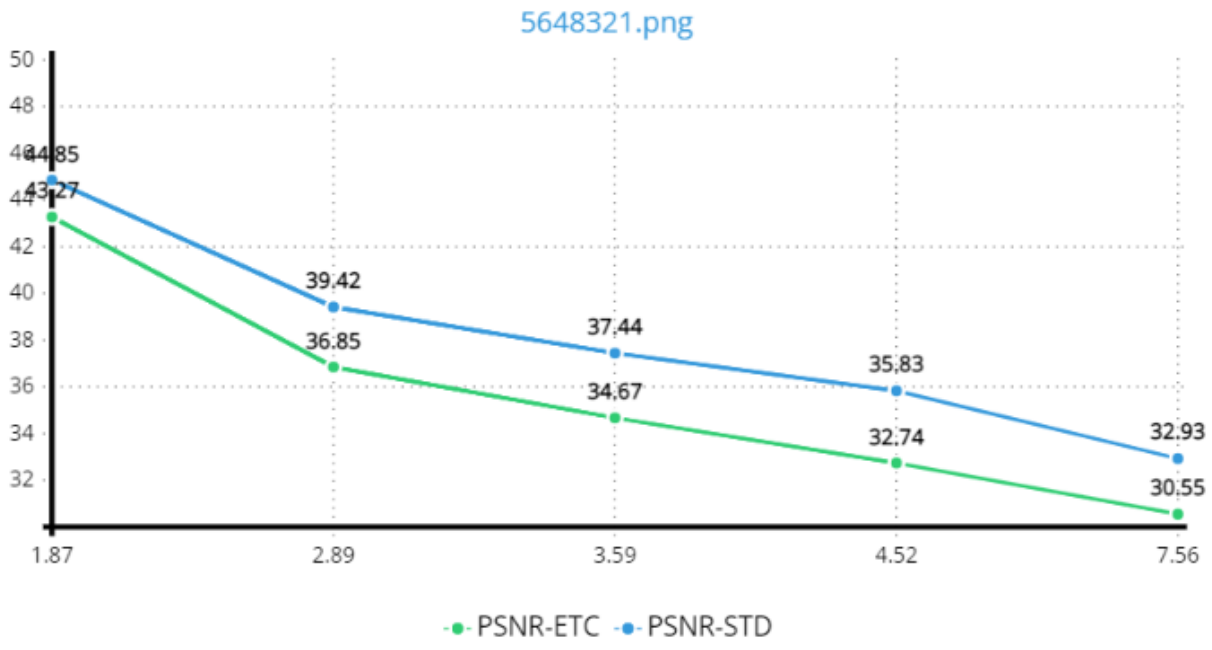
Successivamente, sono stati messi a confronto il PSNR restituito da uno schema standard ed il PSNR restituito dal nostro schema. Nei grafici seguenti, possiamo notare come il valore PSNR del nostro schema si avvicini a quello dello schema standard.




4162702.png
38.82
38.06
33.15
32.01
31.93
30.73
31.02
30.07
29.95
39
38
37
36
35
34
33
32
31
30
2.21
3.52
4.65
6.43
12.52
PSNR-ETC
PSNR-STD


5088155.png
50
45
40
35
30
25
2.17
3.45
4.55
6.09
12.35
PSNR-ETC
PSNR-STD




In particolare, nel grafico riguardante l'immagine "4162702.png" possiamo notare come il PSNR del nostro schema, nonostante il processo di cifratura, risulta essere maggiore rispetto al PSNR dello schema standard di compressione.

# 7. Conclusioni e sviluppi futuri

In conclusione, abbiamo realizzato un nuovo schema di compressione di immagini congiunto ad una fase di crittografia che potrà essere utilizzato assieme al popolare standard di compressione JPEG.
Lo schema proposto risulta efficiente quanto lo schema presentato nel paper di partenza, sia in termini di capacità crittografiche che di qualità di compressione, rendendolo un interessante applicazione pratica come estensione dello standard JPEG.

In aggiunta alla soluzione presentata è stata sviluppata un'alternativa capace di estendere la funzionalità dello schema anche alle immagini a colori, concentrandoci solo sulla parte crittografica omettendo la compressione e considerando immagini in formato PNG.

## *7a. Crittografia di immagini a colori (RGB)*

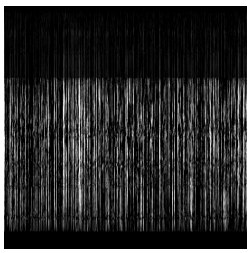
A differenza delle immagini a scala di grigi, le immagini a colori presentano tre componenti (RGB).
La nuova funzionalità dello schema proposto lavora con immagini a colori 512x512, le quali vengono suddivise nelle componenti RGB e, successivamente, ciascuna componente viene crittografata tramite lo stesso procedimento crittografico utilizzato in precedenza, per poi essere inviate singolarmente al destinatario.
Al momento della ricezione, viene effettuata la decrittografia di ciascuna componente per poi effettuare l'unione di queste, ottenendo così l'immagine originale.

*immagine*

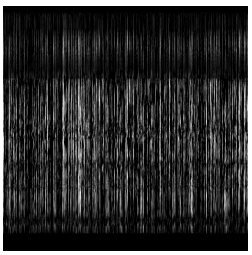
*componente R*

*componente G*

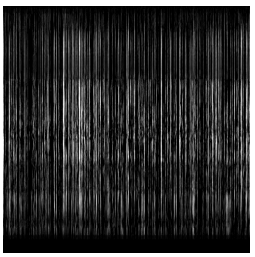
*componente B*



*componente R cifrata*



*componente G cifrata*



*componente B cifrata*

Anche in questo caso, un malintenzionato che è riuscito ad intercettare sul canale di comunicazione le tre componenti crittografate, senza essere in possesso della matrice utilizzata nella fase di cifratura, non sarà in grado di decifrare le componenti e di conseguenza di ottenere l'immagine originale.

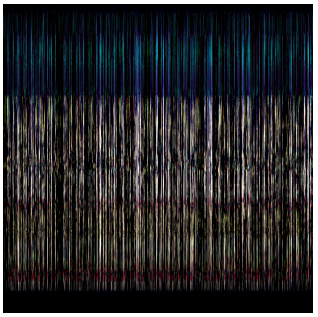


*immagine ottenuta utilizzando una matrice differente*

Un possibile sviluppo futuro consiste nel completare l'adattamento dello schema proposto alle immagini a colori, estendendo questo meccanismo a tutti i tipi di formati esistenti e aggiungendo la parte relativa alla compressione delle immagini.
Inoltre, si potrebbe pensare ad una soluzione che effettua la cifratura senza necessariamente ricorrere alla divisione delle componenti RGB dell'immagine.

# 8. Referenze


1) Dariusz Puchala, Senior Member, IEEE, Kamil Stokfiszewski and Mykhaylo Yatsymirskyy, Member, IEEE. "Encryption Before Compression Coding Scheme for JPEG Image Compression Standard". Institute of Information Technology, Lodz University of Technology, Wolczanska St. 215, 90-924 Lodz, Poland
2) Dariusz Puchala, Senior Member, IEEE, Kamil Stokfiszewski and Mykhaylo Yatsymirskyy, Member, IEEE. "Image Statistics Preserving Encrypt-then-Compress Scheme Dedicated for JPEG Compression Standard". Published online 2021 Mar 31. doi: 10.3390/e23040421
3) https://www.kaggle.com/hsankesara/flickr-image-dataset