

Image-to-Music Steganography Algorithm

Let:

- $I \in Z^{H \times W}$: grayscale cover image of height H , width W
 - $S \in Z^{h \times w}$: secret image (resized)
 - $N = H \times W$: total pixels
-

1. Preprocessing & Resizing

Resize S to match half the width of I and fit height accordingly:

$$w' = \frac{W}{2}, \quad h' = \lfloor w' \cdot \frac{h}{w} \rfloor$$

$$S' \in Z^{h' \times w'} = \text{Resize}(S, w', h')$$

2. Extract Top-4 MSBs

For each pixel $p \in S'$:

$$MSB_4(p) = (p \gg 4) \& 0b1111$$

This gives an array:

$$M = [m_{ij}] \in \{0, 1, \dots, 15\}^{h' \times w'}$$

3. Encoding to Musical Notes → Encoded array

i. Extend each 4-bit value m :

$$m' = m \mid \mid 1001$$

ii. Convert to int (frequency) and map to musical notes :

$$f = \text{int}(m', 2), \quad \text{Note} = \text{librosa.hz_to_note}(f)$$

iii. Convert Note → Hex → Binary via mapping F :

$$\text{Hex} = \text{encode_to_hex}(\text{Note})$$

iv. Replacing the # in all Hex :

$$Hex' = Hex \text{ with } \# \rightarrow 69$$

v. Convert Hex' \rightarrow Binary:

$$Binary = D[Hex']$$

vi. Append '0001' if (Hex' length > 3):

$$Final\ Bits = \{Binary \mid \mid 0001, \text{ if } |Hex'| > 3 \text{ Binary, otherwise}\}$$

vii. Resulting encoded array :

$$E = [e_{ij}] \in \{0, 1, \dots, 15\}^{h' \times 3w'}$$

4. Mapping with Hash-Based Function

i. Define input tuples $T = t_1, t_2, \dots, t_K$ (triplets).

ii. Mapping function using SHA-256:

$$\phi(t) = (SHA256(k \mid \mid t) \bmod 16)$$

Applied row-wise:

$$M' = [\phi(e_{i,j:j+3})]$$

5. Bit Expansion

i. For each symbol $v \in M'$:

$$High(v) = (v \gg 2) \& 0b11$$

$$Low(v) = v \& 0b11$$

$$F = [High(v), Low(v)]$$

ii Final array:

$$F \in \{0, 1, 2, 3\}^{h' \times 2w''}$$

6. Bit Encoding (XOR step)

For every **odd index** i :

$$F_i \leftarrow F_i \oplus 0b11$$

7. Embedding into Cover Image

i. For each pixel $c \in I$:

$$c' = (c \& 0b11111100) \mid (f \& 0b11)$$

ii. Stego-image:

$$I' = \text{Embed}(I, F)$$

Evaluation Metrics

i. *PSNR (Peak Signal-to-Noise Ratio)*

Mean Squared Error (MSE):

$$MSE = \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N (I_1(i, j) - I_2(i, j))^2$$

Where:

- I_1 = Original image (cover)
- I_2 = Stego image
- M, N = Image dimensions

PSNR:

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

Where MAX_I is the maximum pixel value (usually 255).

ii. *SSIM (Structural Similarity Index Measure)*

For two windows x and y :

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

Where:

- μ_x, μ_y = averages of x, y
 - σ_x^2, σ_y^2 = variances
 - σ_{xy} = covariance
 - C_1, C_2 = stability constants
-

iii. BER (Bit Error Rate)

$$BER = \sum_i \frac{1}{n} (b_{o,i} \oplus b_{r,i})$$

$$BER = \frac{\sum_{i=1}^n (b_{o,i} \oplus b_{r,i})}{n}$$

$$BER = \frac{1}{n} \sum_i (b_{o,i} \oplus b_{r,i})$$

Where:

- b_o, b_r = binary arrays of original and received images
 - \oplus = XOR
 - n = total number of bits
-

iv. LPIPS (Learned Perceptual Image Patch Similarity)

$$LPIPS(I, I') = \sum_l w_l \cdot \|\Phi(I) - \Phi(I')\|_2^2$$

Where:

- Φ = feature maps from pretrained networks (e.g., AlexNet)
 - w_l = learned weights
-