Image Steganography using Signal Processing

Mrinashri J
CB.EN.U4ECE22033
Department of Electronics and Communication Engineering
Amrita School of Engineering, Coimbatore
Amrita Viswa Vidyapeetham, India

Priya DharshiniV
CB.EN.U4ECE22040
Department of Electronics and Communication Engineering
Amrita School of Engineering, Coimbatore
Amrita Viswa Vidyapeetham, India

Rosshun S M
CB.EN.U4CCE22045
Department of Electronics and Communication Engineering
Amrita School of Engineering, Coimbatore
Amrita Viswa Vidyapeetham, India

Vijay Venkatesan V
CB.EN.U4ECE22058
Department of Electronics and Communication Engineering
Amrita School of Engineering, Coimbatore
Amrita Viswa Vidyapeetham, India

Abstract—**The aim of our project is to implement image steganography, using its techniques, and show its significance in secure communication, especially unique to the field of medical sciences, primarily focusing on patient data security, confidential image transmission of data such as X-rays, CT scans or MRIs, and preserving electronic health records. We plan to employ this process using Discrete Wavelet Transform (DWT) and the method of Singular Value Decomposition (SVD). DWT is implemented to decompose images into frequency components – approximation, horizontal, vertical, and diagonal details. This decomposition facilitates steganographic embedding within these components. SVD is performed on the DWT output, splitting matrices into constituent parts. These SVD components are manipulated to embed information within the cover image's SVD parameters while ensuring data recovery during the decoding phase for steganography. DWT and SVD are preferred over other methods in this project due to their distinct capabilities such as multiresolution analysis.**

Keywords—image steganography, discrete wavelet transform, singular value decomposition, frequency components – approximation, horizontal, vertical, and diagonal , steganographic embedding

I. Introduction

In today's digital age, secure communication has become increasingly important, particularly in sensitive fields like medical sciences[1]. Patient data privacy and confidentiality are paramount, demanding robust solutions for transmitting sensitive information like X-rays, CT scans, MRIs, and electronic health records. Steganography emerges as a promising technique to address this need.

Steganography is the practice of hiding information within objects so that the concealed data remains undetectable to observers. The term originates from the Greek words 'stegano,' meaning concealed or covered, and 'graphy,' referring to writing or recording. In steganography, confidential messages are concealed within various mediums like images, audio, video, or text files[2]. The inherent nature of medical images, characterized by a wealth of redundant data and intricate details, provides a fertile ground for concealing information[3] within their inherent structure without raising suspicion. This allows for the secure transmission of confidential data, such as diagnoses, treatment plans, and patient identities, through various communication channels, safeguarding patient privacy and ensuring compliance with regulatory requirements.

Within the realm of steganography, which encompasses various techniques for concealing information within different digital media formats like text, audio, and images, image steganography has emerged as a particularly powerful tool for secure communication in the field of medical sciences[4].

By embedding secret data within these mediums, a stego-image or stego-medium is created, appearing outwardly indistinguishable from the original to human perception. Only the sender and receiver can access the hidden data with the help of a password key.

II. Related Work

Ramadhan J. Mustafa [5] explored a range of methods in image steganography, including the least significant bit and S. Thenmozhi, M. Chandrasekaran [6] introduced digital watermarking. They employed a technique called "cropping" to operate on wavelet transfer coefficients for an 8x8 block within the original image. Post message insertion, they implemented a pixel change process to enhance strength. To bolster effectiveness, the authors integrated frequency domain operations. Using the integer wavelet transform, they circumvented issues associated with floating-point accuracy. According to S. Thenmozhi, M. Chandrasekaran (2012), the technique based on the integer wavelet transform yielded adequate results, validated by the peak signal-to-noise ratio.

Paper presented by Savita Bhallamudi[2] implements technique of different LSB algorithm. A real time secret image sharing with fairness where the authors (Lee, G.J. Yong [7]) suggested a technique to hide the validation data at the random coefficient of polynomial to solve difficulty and the time complication. Integer wavelet transform technique is one of the frequently used technique to hide a key while communicating. This technique is proven in the research paper presented by Nagar\m Hamid, Abid Ahaya [8] the technique is robust and secure as no can find the hidden information. Results show very good Peak to signal Noise Ratio of secret information. Reddy, H.S.M., Sathisha, N. and Kumari, A. [9] performed on the steganography using Hybrid Domain Technique and different formats, sizes of cover images are considered to alter to power of 2. The Daubechies Lifting Wavelet Transforms (LWT) is implemented on original image to generate four bands such as XA, XH, XV and XD.

The Fractional Fourier transform (FrFT) which was introduced was initially used for steganography method.

Discrete version of FrFT upgraded into discrete fractional Fourier transform DFrFT and this study is used to compare with other forms of steganography in image processing by (Soni A, Jain J, Roshan R 2013) [10]. In both time domain and frequency domain the values of PSNR are equal but DFrFT gives an additional stego key which is useful for steganography process.

III. Proposed Methodology

The technique used here is to apply discrete wavelet transform to obtain the wavelet coefficient matrices and performing singular value decomposition over them to embed the details of the data image in the cover image. The encoding wavelet used here is "db8" of the Daubechies wavelet for its ability to capture both high-frequency and low-frequency details efficiently, and for analysis across different frequency bands while maintaining a relatively good level of compression.

The dataset is stored in a json file which holds details such as the list of names of the patients, their cover photo, their bio medical detail photo such as their CT-Scan or MRI Scan and the respective passwords for the decoding process to be enabled. Any other data holding files such as csv can also be used. The dataset is loaded into the main file and converted from BGR to RGB formats, and the files are prepared for encoding process.

A. *Mathematical Prelimanaries*

1) *Singular Value Decomposition*

- The SVD of any rectangular matrix produces three unitary matrices U, V and S. U and V are having singular values whereas S singular values are diagonal.

$$\mathbf{A_{nxp}=U_{nxn}\,S_{nxp}\,V^{T}_{pxp}}$$

- For example, a 3×3 matrix and its SVD decomposition are given below. It is visible that matrix A, U, V contains 9 elements but matrix S can be represented by principle diagonal elements because non-diagonal elements are zero.

$$\begin{vmatrix} 1 & 2 & 3 \\ -2 & -3 & 4 \\ 9 & 6 & -1 \end{vmatrix} = \begin{vmatrix} -0.1266 & 0.6722 & 0.7295 \\ 0.3452 & 0.7193 & -0.6029 \\ -0.9300 & 0.1755 & -0.3231 \end{vmatrix} \begin{vmatrix} 11.6318 & 0 & 0 \\ 0 & 4.7937 & 0 \\ 0 & 0 & 1.6499 \end{vmatrix} \begin{vmatrix} -0.7898 & 0.1696 & 0.1696 \\ -0.5905 & 0.0499 & 0.8055 \\ 0.1660 & 0.9843 & 0.0607 \end{vmatrix}$$

Fig. 1. Matrix Representaiton

- So, instead of sending a complete message image the proposed technique sending only the S matrix and complete message image could reconstruct using USV matrices. Furthermore, when the image is somewhat disrupted, the diagonal elements of the singular value matrix acquired from singular-value decomposition do not vary appreciably, indicating that the singular-value decomposition of the matrix is rotation invariant. In this regard, adding the secret image after the image's singular value decomposition can boost the algorithm's anti-attack performance

2) *Discrete Wavelet Transform*

- DWT is a very popular mathematical tool in the field of signal analysis to decompose any image in low (LL) and high (HH) components.
- The Low-frequency component contains more information and the high component contains less information. So, any change in a cover image due to data hiding in the HH component causes less distortion in the cover image. HH component of DWT matrix will hold the secret data.
- DWT has several advantages, including the ability to withstand multiple attacks while maintaining the image's original quality and ensuring the integrity of secret information collected.

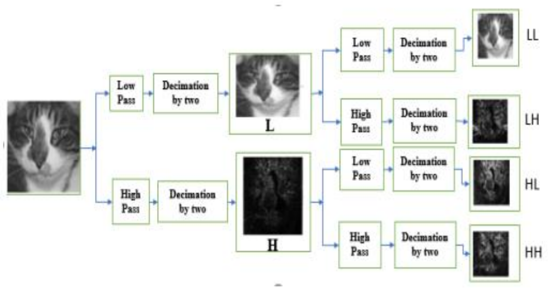


Fig. 2. DFT Algorithm

B. *Encoding Process*

1) *Splitting the Red, Green and Blue channels*

- The three main channels of the images are split and stored in three different arrays which are two dimensional.
- The respective arrays contain pixel values representing the respective colour intensities of each pixel in the image. Each value in this array denotes the amount of red colour present at the corresponding pixel location.
- This is done for both the cover and data to be hidden images.

2) *Apply DWT to the obtained channels*

- DWT is applied to the respective red, green and blue channel matrices and the corresponding coefficients for approximation, horizontal and vertical details are obtained.
- This is done for both the cover and data to be hidden images.

3) *Perform SVD over the DWT coefficient matrices*

- SVD is applied over the respective red, green, and blue channel DWT coefficient matrices to obtain the corresponding left singular vectors matrix, the singular values array and right singular vectors matrix.
- The vectors form an orthonormal basis for column space of original matrix which represent a direction in the input space, a singular value 1-D array that describes the "importance" or "strength" of each corresponding singular vector and a right singular value matrix with vectors that form an orthonormal basis for the row space of the original matrix which corresponds to a direction in the output space, respectively.
- This is done for both the cover and data to be hidden images.

4) *Embed the information and reconstruct the matrix*

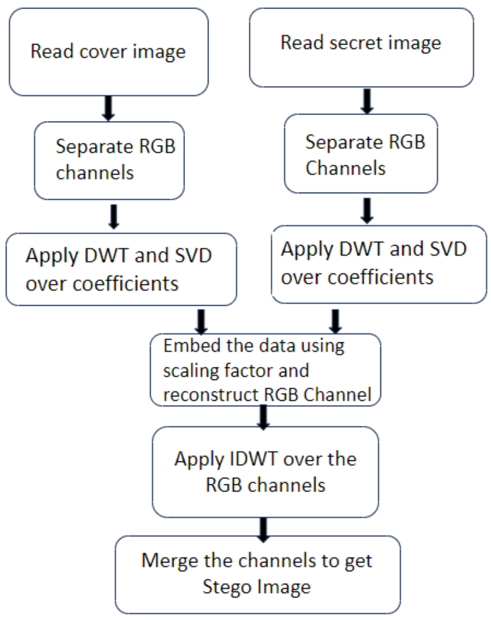
- Embed the information of the data image to the cover image from the singular value array by multiplying with a scaling factor which determines how much of the image is to be embedded. A higher scaling factor would embed more information, but image quality will be lost.

- Apply these for all three channels (red, green and blue)
- Reconstruct the coefficient matrix from the embedded SVD parameters by scalar matrix multiplication.
- Perform type casting on the reconstructed coefficients for each colour channel convert the data type of each element in the matrix from its current type to integers.
- Merge the three channels to form a single image matrix.

5) Extract details and apply IDWT to generate steganographic image

- Split the processed image into its individual colour channels and pair each channel with its corresponding detail coefficients and store it in a tuple.
- The resulting tuple is taken, and inverse DWT is performed. It reconstructs the image data from the wavelet coefficients, thereby reversing the process of wavelet decomposition.
- The three reconstructed colour channels are merged into a single multi-channel image matrix which is the final steganographic image.

Flowchart for the encoding process is as follows:


Read cover image
Read secret image
Separate RGB channels
Separate RGB Channels
Apply DWT and SVD over coefficients
Apply DWT and SVD over coefficients
Embed the data using scaling factor and reconstruct RGB Channel
Apply IDWT over the RGB channels
Merge the channels to get Stego Image

C. Decoding Process

1) Check if the password entered by user for the patient and proceed to decode only if the password is correct

2) Apply DWT on the stego image

- Apply DWT as in encoding to obtain the approximate, horizontal, vertical and diagonal coefficient details for all three channels (red, green and blue)

3) Perform SVD over the coefficient matrices

- Obtain the left singular value matrices, singular value 1-D array and right singular value matrices.

4) Reverse the information embedded in the 'D' parameter of the cover image that was performed in encoding process

5) Combine the approximations with the hidden SVD values to reconstruct the hidden image

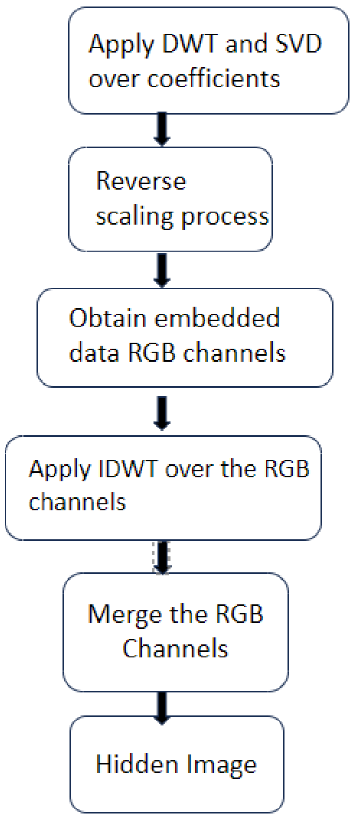
6) Pair each colour channel's pixel values with its respective DWT coefficients in a tuple to perform IDWT

7) Apply inverse transform to each channel of the image to generate the final hidden information image

- Hidden stego images get high resolution R,G,B separate images/channels using IDWT

8) Combine the different high resolution r,g,b to get the final hidden information

Flowchart for the decoding process is as follows:


Apply DWT and SVD over coefficients
Reverse scaling process
Obtain embedded data RGB channels
Apply IDWT over the RGB channels
Merge the RGB Channels
Hidden Image

IV. Results and Discussions

A. Dataset description

This work has been implemented in Python. The dataset required for this work are the list of patient names, their photos which will act as the cover image and the photo of their medical data which will be the image to be hidden. This dataset is collected[11] and stored in a json file for loading into the main file. This can be done for a huge dataset collected; here we have tested the performance metrics for a sample of 3 patients with various scaling factor differences between encoding and decoding process to determine accuracy in retrieving the hidden image back from the stego image.

B. *Performance metrics in a tabular form*

TABLE I. PATIENT NAME WITH SCALING FACTOR USED

S. No	**Performance Metrics**		
	Patient Name	***Scaling Factor Difference***	***Image Quality***
1	Proakis	0.001	Darkened and distorted
2	Simon	0.1	Retained
3	Shreya	5	Brightened and distorted

C. *Simulation results*

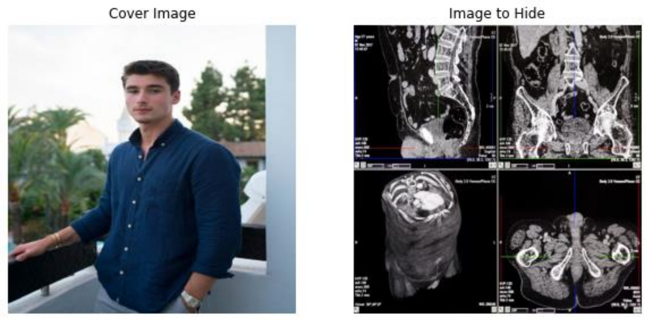


Fig. 3. Proakis's cover image and data image

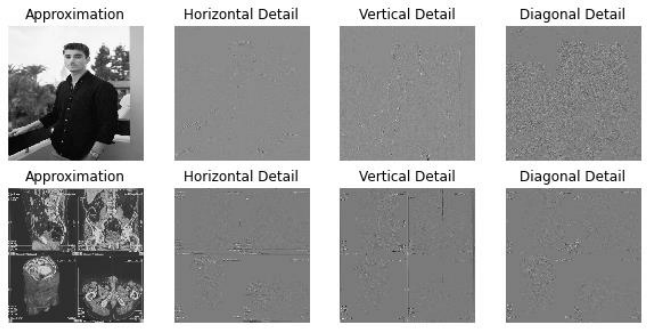


Fig. 4. Wavelet coefficient details after applying transform with scaling factor = 0.001

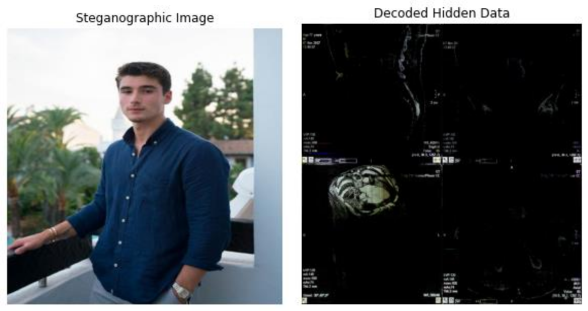


Fig. 5. Stego image and the decoded secret image

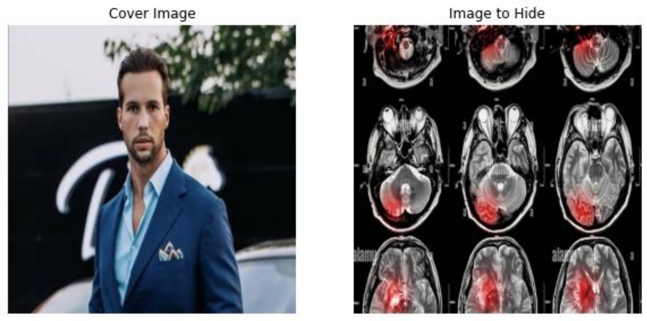


Fig. 6. Simon's cover image and data image

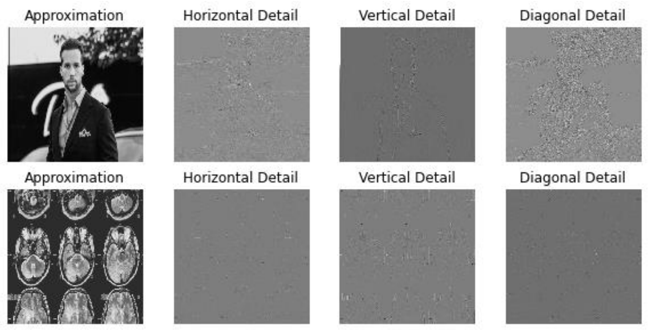


Fig. 7. Wavelet coefficient details after applying transform with scaling factor = 0.1

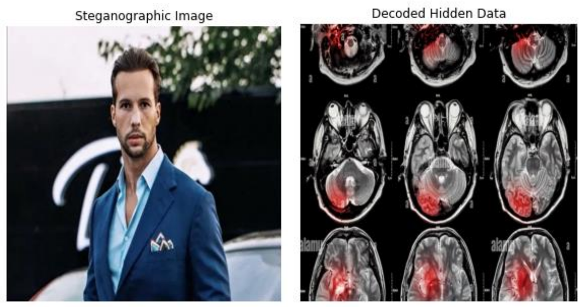


Fig. 8. Stego image and the decoded secret image

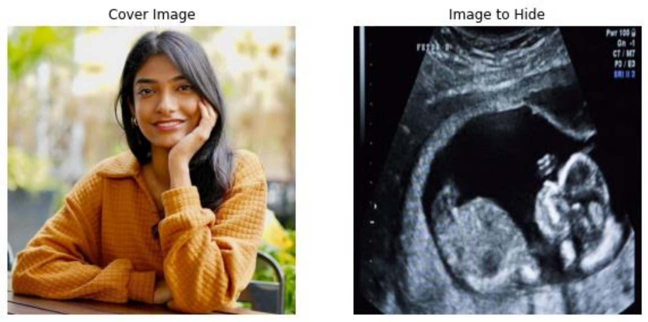


Fig. 9. Shreya's cover image and data image

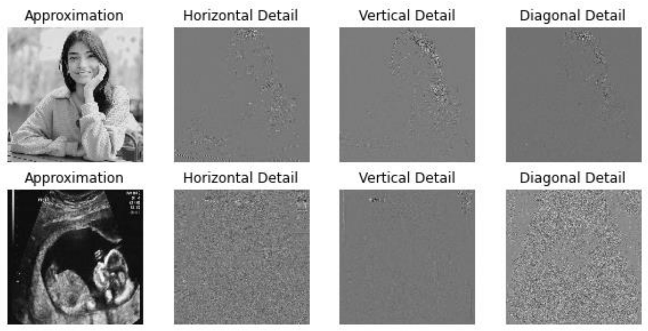


Fig. 10. Wavelet coefficient details after applying transform with scaling factor = 5

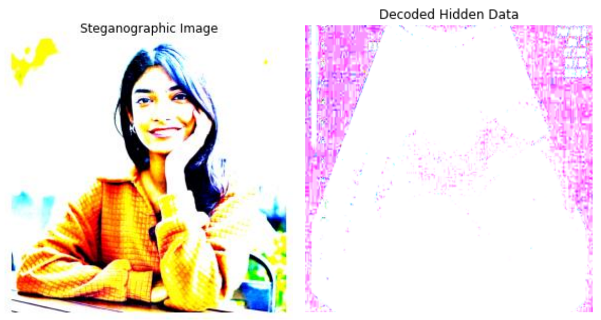


Fig. 11. Stego image and the decoded secret image

TABLE:				
	Alpha	SSIM	PSNR	MSE
0	0.005	0.999989	56.387634	0.1493
1	0.010	0.999961	50.912738	0.5269
2	0.015	0.999897	46.725284	1.3821
3	0.020	0.999800	43.825817	2.6946
4	0.025	0.999676	41.725152	4.3708
5	0.030	0.999518	39.978591	6.5346
6	0.035	0.999326	38.504770	9.1749
7	0.040	0.999102	37.244227	12.264720
8	0.045	0.998848	36.142967	15.804605
9	0.050	0.998562	35.163875	19.801260
10	0.055	0.998247	34.286393	24.234888
11	0.060	0.997902	33.489466	29.116137
12	0.065	0.997527	32.757791	34.458884
13	0.070	0.997123	32.082645	40.254634
14	0.075	0.996690	31.456996	46.492274
15	0.080	0.996229	30.872765	53.186804
16	0.085	0.995740	30.325484	60.329808
17	0.090	0.995222	29.809017	67.948307
18	0.095	0.994678	29.323001	75.994169
19	0.100	0.994106	28.862241	84.499916
20	0.105	0.993508	28.424464	93.461755
21	0.110	0.992884	28.008231	102.862549
22	0.115	0.992234	27.610935	112.716343
23	0.120	0.991559	27.230515	123.035064
24	0.125	0.990857	26.865720	133.816088
25	0.130	0.990130	26.515446	145.056020
26	0.135	0.989381	26.179616	156.717955
27	0.140	0.988608	25.856405	168.826204
28	0.145	0.987810	25.544546	181.395170
29	0.150	0.986989	25.243279	194.425140
30	0.155	0.986145	24.952122	207.906536
31	0.160	0.985277	24.669947	221.863409
32	0.165	0.984388	24.397129	236.247637
33	0.170	0.983477	24.132455	251.093124
34	0.175	0.982545	23.875923	266.371666
35	0.180	0.981590	23.626351	282.127396
36	0.185	0.980615	23.383942	298.322570
37	0.190	0.979620	23.148271	314.958471
38	0.195	0.978604	22.918602	332.062752
39	0.200	0.977568	22.694863	349.618190

D. *Inference*

From the results obtained from the simulation it is clear that when the difference between the scaling factor in encoding and decoding process is minimal, the quality of the stego and secret image are retained and when the scaling factor is too low, the image is darkened and when the scaling factor is too high, the exposure of the image increases drastically and the quality of the image is very poor.

V. Conclusion

Image steganography stands out as an intriguing and captivating technique for concealing data, having a rich historical application. The initial phase involves raising awareness about the existence of such methods, particularly for safeguarding highly sensitive information such as passwords or the transfer of medical-related data as taken up in this paper. It can complement watermarking and various storage mechanisms. Enhanced familiarity with its capabilities and versatility would facilitate its smoother adoption and utilization.

This paper explores the use of the Discrete Wavelet Transform (DWT) employing Singular Value Decomposition (SWD) to embed an image within another image file. However, two major limitations of this method are the possibilities of losing image quality if the scaling factor is not set properly and the necessity of making the sizes of the image the same before applying DWT and SVD to embed them properly.

Additionally, this method can be further be developed using advanced algorithms to embed multiple secret images, or even other data types such as audio or video signals.

VI. References

[1] M. Elhoseny, G. Ramírez-González, O. M. Abu-Elnasr, S. A. Shawkat, N. Arunkumar and A. Farouk, "Secure Medical Data Transmission Model for IoT-Based Healthcare Systems," in IEEE Access, vol. 6, pp. 20596-20608, 2018, doi: 10.1109/ACCESS.2018.2817615.

[2] Bhallamudi, Savitha. (2015). Image Steganography. 10.13140/RG.2.2.21323.18727.

[3] G. Vallathan, G. G. Devi and A. V. Kannan, "Enhanced data concealing technique to secure medical image in telemedicine applications," 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, India, 2016, pp. 186-190, doi: 10.1109/WiSPNET.2016.7566117.

[4] N. Subramanian, O. Elharrouss, S. Al-Maadeed and A. Bouridane, "Image Steganography: A Review of the Recent Advances," in IEEE Access, vol. 9, pp. 23409-23423, 2021, doi: 10.1109/ACCESS.2021.3053998

[5] Ramadhan J Mstafa, "Image steganography using discrete fractional Fourier transform," ASEE,2013 Northeast Section Conference on, 14- 16 March 2013

[6] S.Thenmozhi,M.Chandrasekaran, "Novel approach for image stenograpgy based on integer wavelet transform," 2012 IEEE

International Conference on Computer Intelligence and Computing Research ,2012,pp.1-5,doi: 10.1109/ICCIC.2012.6510300.

[7] Lee G.J, Yoon E.J, Kim C, Yong, K.Y, “ A real-time secret image sharing with fairness,”2016.[Online].

[8] Nagam hamid, Abid Ahaya, R. Badlishah, “Image Steganography Overview: Techniques,” International Journal of Computer Science and Security (IJCSS), Vol., no., pp.6,3, 2012

[9] Reddy H.S.M., Sathisha N, Kumari A, Raja K.B, "Secure steganography using hybrid domain technique," Computing Communication & Networking Technologies (ICCCNT), 2012 Third International Conference on, vol., no., pp.1,11, 26-28 July 2012

[10] Soni A,Jain J, Roshan R., "Image steganography using discrete fractional Fourier transform," Intelligent Systems and Signal Processing (ISSP), 2013 International Conference on, vol., no., pp.97,100, 1-2 March 2013

[11] Angel Cruz-Roa - Image Datasets. (n.d.). https://sites.google.com/site/aacruzr/image-datasets

VII. Appendix

Code implementation in Python:

A. Main code

```
import numpy as np
import pywt
import matplotlib.pyplot as plt
import cv2
import json

def read_and_convert_image(image_path):
   image = cv2.imread(image_path)  #to read the image
file
   image = cv2.cvtColor(image,
cv2.COLOR_BGR2RGB)  #converting BGR to RGB
   return image

def dwt_coefficients(image, encoding_wavelet):
   cA, (cH, cV, cD) = pywt.dwt2(image,
encoding_wavelet)
   return cA, cH, cV, cD

def show_coefficients_subplot(coefficients, subplot_base,
dwt_labels):
   for i, a in enumerate(coefficients):
      subplot_number = subplot_base + i
      plt.subplot(4, 4, subplot_number)
      plt.title(dwt_labels[i])
      plt.axis('off')
      plt.imshow(a, interpolation="nearest",
cmap=plt.cm.gray)

def svd_decomposition(matrix):
   P, D, Q = np.linalg.svd(matrix, full_matrices=False)
   return P, D, Q

# Function to validate the password for decoding
def validate_password(patient_choice, entered_password,
patients_data):
   if patient_choice in patients_data:
      patient_info = patients_data[patient_choice]
      return entered_password == patient_info['Password']
   return False

encoding_wavelet = "db8"  # indicates which waveform
(wavelet) was used for the wavelet transform
decoding_wavelet = "db8"

def load_patient_data(json_file):
   with open(json_file, 'r') as file:
      data = json.load(file)
   return data

# Load patient data from JSON
patients_data = load_patient_data('patient_details.json')

# Ask for patient choice and validate password
while True:
   patient_choice = input("\nWhich patient's medical data
do you wish to see? (Proakis, Simon, Shreya): ")

   if patient_choice in patients_data:
      data_image_path =
patients_data[patient_choice]['Image_to_Hide']
      cover_image_path =
patients_data[patient_choice]['Image_to_Send']

      data_image =
read_and_convert_image(data_image_path)
      cover_image =
read_and_convert_image(cover_image_path)

      plt.figure(figsize=(10, 5))

      plt.subplot(1, 2, 1)
      plt.axis("off")
      plt.title("Cover Image")
      plt.imshow(cover_image, aspect="equal")

      plt.subplot(1, 2, 2)
      plt.axis("off")
      plt.title("Image to Hide")
      plt.imshow(data_image, aspect="equal")

      plt.show()

      # -------------------------------------------
      # Encoding Process
      # Process of hiding an image within another image
      # --------------------------------------------

      #Separating channels (colors) for cover and hidden
images
      cover_image_r = cover_image[:, :, 0] #extracting the
red channel for cover image
      cover_image_g = cover_image[:, :, 1] #extracting the
green channel for cover image
      cover_image_b = cover_image[:, :, 2] #extracting the
blue channel for cover image

      data_image_r = data_image[:, :, 0] #extracting the
red channel for data image
      data_image_g = data_image[:, :, 1] #extracting the
green channel for data image
      data_image_b = data_image[:, :, 2] #extracting the
blue channel for data image

      # Apply the wavelet transform to the cover and
hidden images
      cAr, cHr, cVr, cDr =
dwt_coefficients(cover_image_r, encoding_wavelet) #gets
```

```
the approximation, horizontal, vertical and diagonal coefficients for red channel of cover image
    cAg, cHg, cVg, cDg = dwt_coefficients(cover_image_g, encoding_wavelet) #gets the approximation, horizontal, vertical and diagonal coefficients for green channel of cover image
    cAb, cHb, cVb, cDb = dwt_coefficients(cover_image_b, encoding_wavelet) #gets the approximation, horizontal, vertical and diagonal coefficients for blue channel of cover image

    cAr1, cHr1, cVr1, cDr1 = dwt_coefficients(data_image_r, encoding_wavelet) #gets the approximation, horizontal, vertical and diagonal coefficients for red channel of data image
    cAg1, cHg1, cVg1, cDg1 = dwt_coefficients(data_image_g, encoding_wavelet) #gets the approximation, horizontal, vertical and diagonal coefficients for green channel of data image
    cAb1, cHb1, cVb1, cDb1 = dwt_coefficients(data_image_b, encoding_wavelet) #gets the approximation, horizontal, vertical and diagonal coefficients for blue channel of data image

    # Plot all layers resulted from DWT
    dwt_labels = ["Approximation", "Horizontal Detail", "Vertical Detail", "Diagonal Detail"]

    plt.figure(figsize=(10, 10))

    #plotting the coefficient details
    show_coefficients_subplot([cAr, cHr, cVr, cDr], 5, dwt_labels)
    show_coefficients_subplot([cAr1, cHr1, cVr1, cDr1], 9, dwt_labels)

    # Perform Singular Value Decomposition (SVD) on the cover and hidden images

    #P: Left singular value matrix vectors that form an orthonormal basis for the column space of the original matrix
    #Each column vector in P represents a direction in the input space.
    #D: Singular values 1-D array that describes the "importance" or "strength" of each corresponding singular vector in P and Q
    #Q: Right singular value matrix vectors that form an orthonormal basis for the row space of the original matrix
    #Each row vector in Q corresponds to a direction in the output space.

    Pr, Dr, Qr = svd_decomposition(cAr)
    Pg, Dg, Qg = svd_decomposition(cAg)
    Pb, Db, Qb = svd_decomposition(cAb)

    P1r, D1r, Q1r = svd_decomposition(cAr1)
    P1g, D1g, Q1g = svd_decomposition(cAg1)
    P1b, D1b, Q1b = svd_decomposition(cAb1)

    # Embed the hidden information into the 'D' parameters of the cover image
    # The singular values are embedded by using a scaling factor
    # More the scaling factor: better the embedding but image quality will be lost
    # So an ideal value of 0.1 is chosen here
    Embed_Dr = Dr + (0.10 * D1r)
    Embed_Dg = Dg + (0.10 * D1g)
    Embed_Db = Db + (0.10 * D1b)

    # Reconstruct the coefficient matrix from the embedded SVD parameters
    # by scalar matrix multiplcation
    imgr = np.dot(Pr * Embed_Dr, Qr)
    imgg = np.dot(Pg * Embed_Dg, Qg)
    imgb = np.dot(Pb * Embed_Db, Qb)

    # Converting the image to int 0-255 range from float resulted in SVD computation
    a = imgr.astype(int)  # red matrix
    b = imgg.astype(int)  # green matrix
    c = imgb.astype(int)  # blue matrix

    # Concatenate the three reconstructed RGB channels into a single matrix
    img = cv2.merge((a, b, c))

    # Extract the horizontal, vertical, and diagonal coefficients from each RGB channel of the image
    # to recreate a original img but with cA now having hidden info
    # cHr, cVr, cDr represet the wavelet coefficients corresponding to the horizontal (cHr), vertical (cVr), and diagonal (cDr) components
    # obtained from the wavelet transform of the red channel
    proc_r = img[:, :, 0], (cHr, cVr, cDr)  # extracts pixel values only from the red channel of the image
    proc_g = img[:, :, 1], (cHg, cVg, cDg)  # extracts pixel values only from the green channel of the image
    proc_b = img[:, :, 2], (cHb, cVb, cDb)  # extracts pixel values only from the blue channel of the image

    # Apply inverse transform to each channel of the processed image, generating the steganographic image
    # It reconstructs the image data from the wavelet coefficients, thereby reversing the process of wavelet decomposition
    processed_rgbr = pywt.idwt2(proc_r, encoding_wavelet)
    processed_rgbg = pywt.idwt2(proc_g, encoding_wavelet)
    processed_rgbb = pywt.idwt2(proc_b, encoding_wavelet)

    # The three reconstructed color channels are merged into a single multi-channel image matrix
    finalimg = cv2.merge((processed_rgbr.astype(int), processed_rgbg.astype(int), processed_rgbb.astype(int)))

    # Plot steganographic image
    plt.figure(figsize=(5, 5))
    plt.axis('off')
    plt.title("Steganographic Image")
    plt.imshow(finalimg, aspect="equal")
    plt.show()
```

--
# Decoding Process
# Steganography reversal process

# Check if the password is correct before proceeding with the decoding process
entered_password = input(f"\nEnter the password for {patient_choice}: ")

if validate_password(patient_choice, entered_password, patients_data):

# Apply the decoding transform to each channel of the stego image
# applying dwt to 3 stego channel images to get coeffs of stego image in r,g,b
# Decompose the image data into approximations (low-frequency) and detail coefficients (high-frequency)
Psend_r = pywt.dwt2(processed_rgbr, decoding_wavelet)
PcAr, (PcHr, PcVr, PcDr) = Psend_r # Get the approximate, horizontal, vertical and diagonal detail coefficients for red channel as a tuple where data may be embedded
Psend_g = pywt.dwt2(processed_rgbg, decoding_wavelet)
PcAg, (PcHg, PcVg, PcDg) = Psend_g # Get the approximate, horizontal, vertical and diagonal detail coefficients for green channel as a tuple where data may be embedded
Psend_b = pywt.dwt2(processed_rgbb, decoding_wavelet)
PcAb, (PcHb, PcVb, PcDb) = Psend_b # Get the approximate, horizontal, vertical and diagonal detail coefficients for blue channel as a tuple where data may be embedded

dwt_labels = ["Approximation", "Horizontal Detail", "Vertical Detail", "Diagonal Detail"]

plt.figure(figsize=(10, 10))

#plotting the coefficient details
show_coefficients_subplot([PcAr, PcHr, PcVr, PcDr], 5, dwt_labels)

# Perform Singular Value Decomposition (SVD) on the stego image
# PP: Left singular value matrix vectors
# PD: Singular values 1-D array that describes the "importance" or "strength" of each corresponding singular vector in P and Q
# PQ: Right singular value matrix vectors
# full_matrices = False gives compact form of SVD for memory efficiency
PPr, PDr, PQr = np.linalg.svd(PcAr, full_matrices=False)
PPg, PDg, PQg = np.linalg.svd(PcAg, full_matrices=False)
PPb, PDb, PQb = np.linalg.svd(PcAb, full_matrices=False)

# Reverse the information embedded in the 'D' parameter of the cover image that was performed in encoding process
# Subtract from R,G,B channels values of cover image
Decode_r = (PDr - Dr) / 0.10
Decode_g = (PDg - Dg) / 0.10
Decode_b = (PDb - Db) / 0.10

# Combine the approximations with the hidden SVD values to reconstruct the hidden image
# Calculating normalized differences between the singular values obtained during the decoding phase and the singular values obtained during the encoding phase for the red, green, and blue channels, respectively.
reimgr = np.dot(P1r * Decode_r, Q1r)
reimgg = np.dot(P1g * Decode_g, Q1g)
reimgb = np.dot(P1b * Decode_b, Q1b)

# Obtain the reconstructed hidden image, which consists of the color channels combined with the normalized SVD differences
# Converting the float point numbers to integers for calculation for red, green and blue channels respectively
d = reimgr.astype(int)
e = reimgg.astype(int)
f = reimgb.astype(int)
merged_img = cv2.merge((d, e, f)) # To create a single color image by combining these three channels

# Pair each colour channel's pixel values with its respective DWT coefficients in a tuple to perform IDWT
proc_r = merged_img[:, :, 0], (cHr1, cVr1, cDr1) # Extracts pixel values only from the red channel of the image
proc_g = merged_img[:, :, 1], (cHg1, cVg1, cDg1) # Extracts pixel values only from the green channel of the image
proc_b = merged_img[:, :, 2], (cHb1, cVb1, cDb1) # Extracts pixel values only from the blue channel of the image

# Apply inverse transform to each channel of the image to generate the final hidden information image
# Hidden stego images get high resolution r,g,b seperate images/channels using IDWT
processed_rgbr = pywt.idwt2(proc_r, decoding_wavelet)
processed_rgbg = pywt.idwt2(proc_g, decoding_wavelet)
processed_rgbb = pywt.idwt2(proc_b, decoding_wavelet)

# Converting float to int for cv2 module to handle merging of pixel values
x1 = processed_rgbr.astype(int) # For red channel
y1 = processed_rgbg.astype(int) # For green channel
z1 = processed_rgbb.astype(int) # For blue channel

# Combine different high resolution r,g,b to get hidden image

```
            hidden_image = cv2.merge((x1, y1, z1))

            # Display the decoded hidden image
            plt.figure(figsize=(5, 5))

            plt.axis('off')
            plt.title("Decoded Hidden Data")
            plt.imshow(hidden_image, aspect="equal")
            plt.show()

            break

        else:
            print("Incorrect password. Access denied. Program
Terminating")
            break
    else:
        print("Invalid patient choice. Try again...")
```

B. Sample dataset in a json file

```
{
    "Proakis": {
        "Image_to_Hide": "patient1_to_hide.png",
        "Image_to_Send": "patient1_to_send.png",
        "Password": "parseval"
    },
    "Simon": {
        "Image_to_Hide": "patient2_to_hide.png",
        "Image_to_Send": "patient2_to_send.png",
        "Password": "euler"
    },
    "Shreya": {
        "Image_to_Hide": "patient3_to_hide.png",
        "Image_to_Send": "patient3_to_send.png",
        "Password": "wavelet"
    }
```