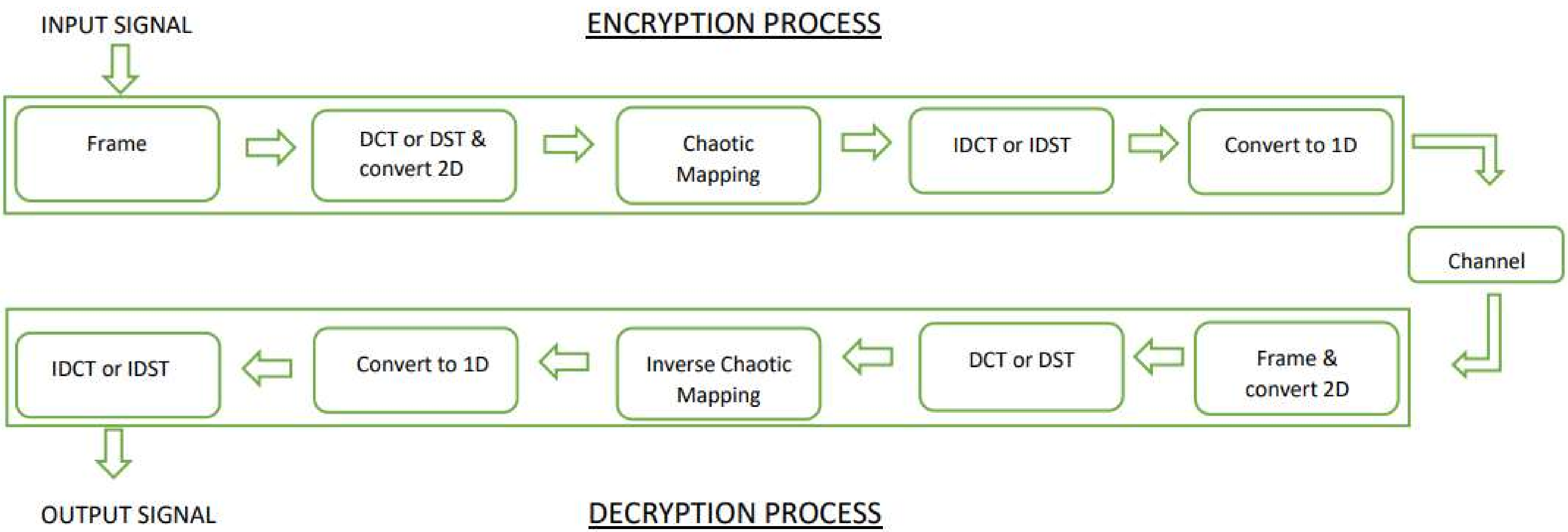
Speech Encryption and Decryption



References:

Speech Encryption Using Two Dimensional Chaotic Maps - D Alzharaa Mostafa *, Naglaa. F. Soliman *, Mohamoud Abdalluh* Fathi E. Abd El-samieD

Encryption/Decryption System

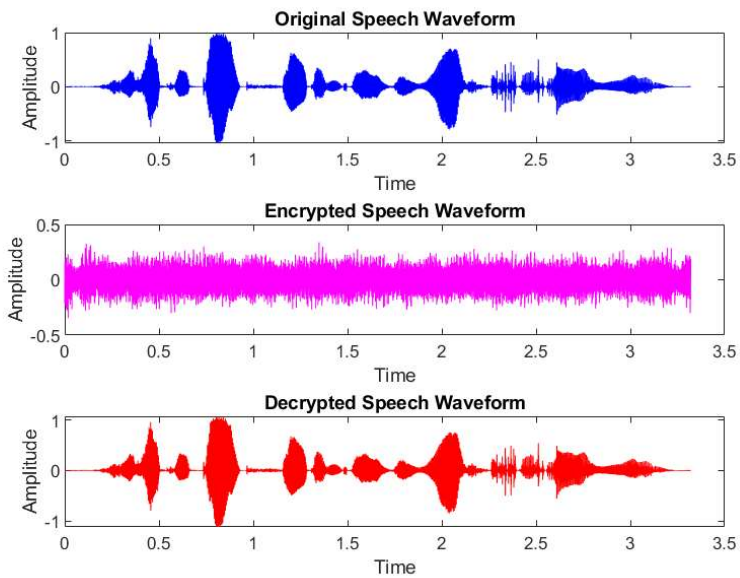
```
close all; clc; clear all;

% choosing only .wav files from one directory of TIMIT Dataset
files = dir(fullfile('C:\Users\Amrut Biju\Desktop\Study\6th Semester\SP\Project\Code\TIMIT\TRAIN\DR8\FCLT0\', '*.wav'));
L = length(files); % no. of .wav files
L = 5;

% will contain all the filepaths of the .wav files
waves = {};

for i=1:L
    file=files(i).name;
    filepath = fullfile( 'C:\Users\Amrut Biju\Desktop\Study\6th Semester\SP\Project\Code\TIMIT\TRAIN\DR8\FCLT0\', file );
    waves{end+1} = filepath;
end
```

```
original_speech_waveforms = {};
```

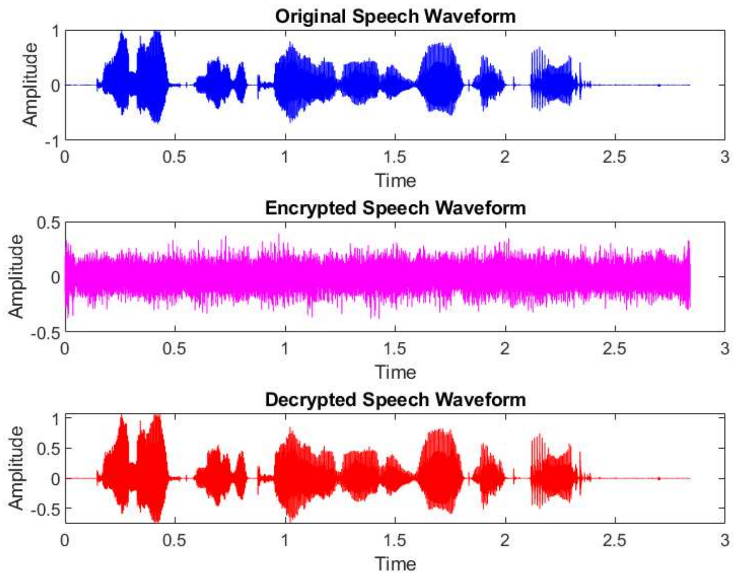
```
encrypted_speech_waveforms = {};
decrypted_speech_waveforms = {};

% iterating through each .wav file
for i = 1:length(waves)
    % reading the current .wav file
    [sample_sound,fs]=audioread(waves{i});

    fprintf('audio sample number : %d', i);

    % normalizing
    sample_sound = sample_sound./max(sample_sound);

    % frame size = 20 ms, frame shift = 10 ms
    frame_size = 0.020;
    frame_shift = 0.010;
    frame_length = frame_size*fs; % no. of samples in one frame (here it is 200 samples for 25 ms frame size, fs = 8000 Hz)

    % padding with zeros
    if mod(length(sample_sound), frame_length) ~= 0
        sample_sound = [sample_sound; (zeros(1, frame_length-mod(length(sample_sound), frame_length)))'];
    end

    % plotting the original sample waveform
    figure
    subplot(311)
    plot((0:length(sample_sound)-1)/fs,sample_sound, 'b');
    xlabel("Time");
    ylabel("Amplitude");
    title('Original Speech Waveform');
    original_speech_waveforms{end+1} = sample_sound;
    sound(sample_sound, fs);
    pause(5);

%     filename = sprintf('Audiofile_%d_original_speech.wav', i);
%     audiowrite(filename,sample_sound,fs)

    % Encryption
    [Encrypted_speech, Chaotic_map] = encryption(sample_sound, fs);

    % plotting the encrypted speech waveform
    subplot(312)
    plot((0:length(Encrypted_speech)-1)/fs,Encrypted_speech, 'm');
    xlabel("Time");
    ylabel("Amplitude");
    title('Encrypted Speech Waveform');
    encrypted_speech_waveforms{end+1} = Encrypted_speech;
    sound(Encrypted_speech, fs);
    pause(5);

%     filename = sprintf('Audiofile_%d_encrypted_speech.wav', i);
%     audiowrite(filename,Encrypted_speech,fs);

    % Decryption
```

```
    [Decrypted_speech] = decryption(Encrypted_speech, Chaotic_map, fs);

    % plotting the decrypted speech waveform
    subplot(313)
    plot((0:length(Decrypted_speech)-1)/fs,Decrypted_speech, 'r');
    xlabel("Time");
    ylabel("Amplitude");
    title('Decrypted Speech Waveform');
    decrypted_speech_waveforms{end+1} = Decrypted_speech;
    sound(Decrypted_speech, fs);
    pause(5);

%     filename = sprintf('Audiofile_%d_decrypted_speech.wav', i);
%     audiowrite(filename,Decrypted_speech,fs);
end
```

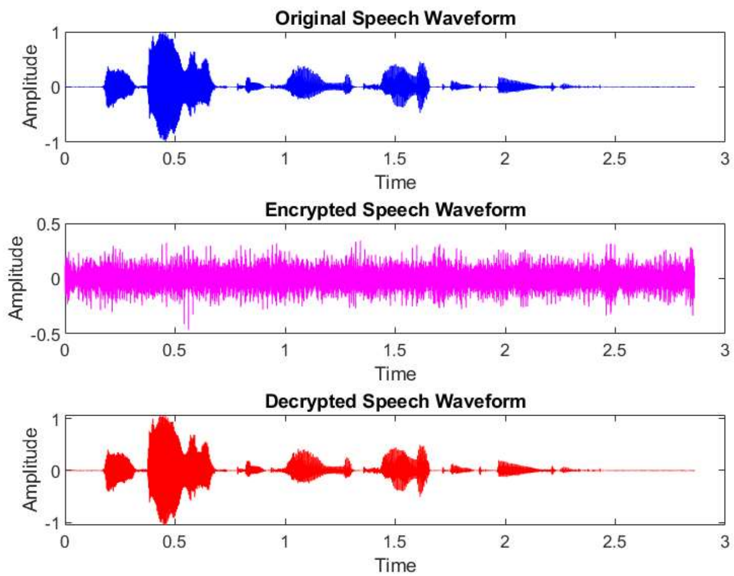
audio sample number : 1



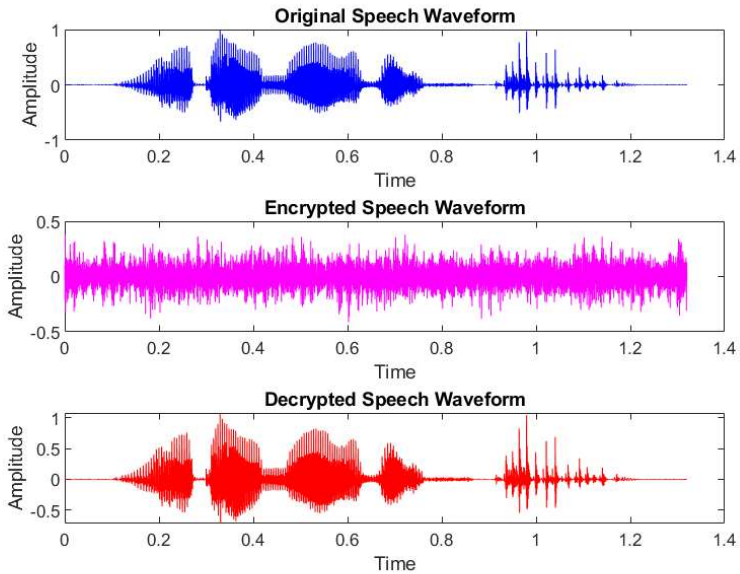
audio sample number : 2



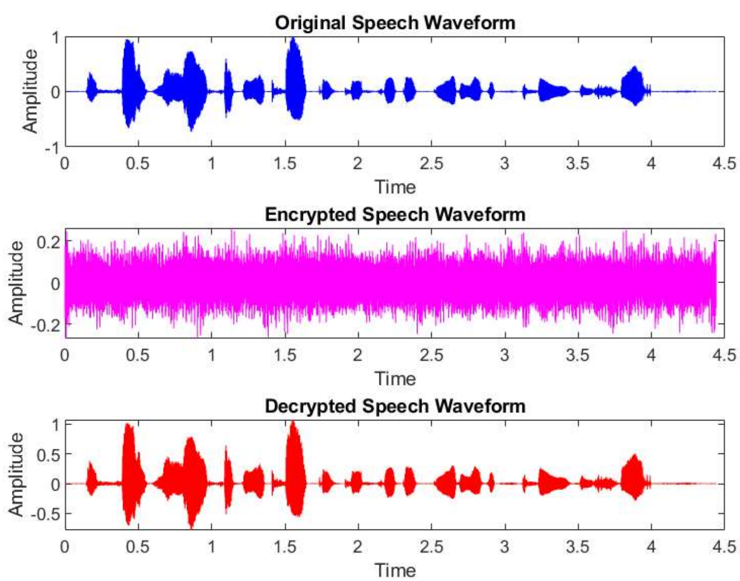
audio sample number : 3



audio sample number : 4




```
audio sample number : 5
```



<u>Evaluating our System:</u>

<u>1. Using cross correlation and correlation coefficients:</u>

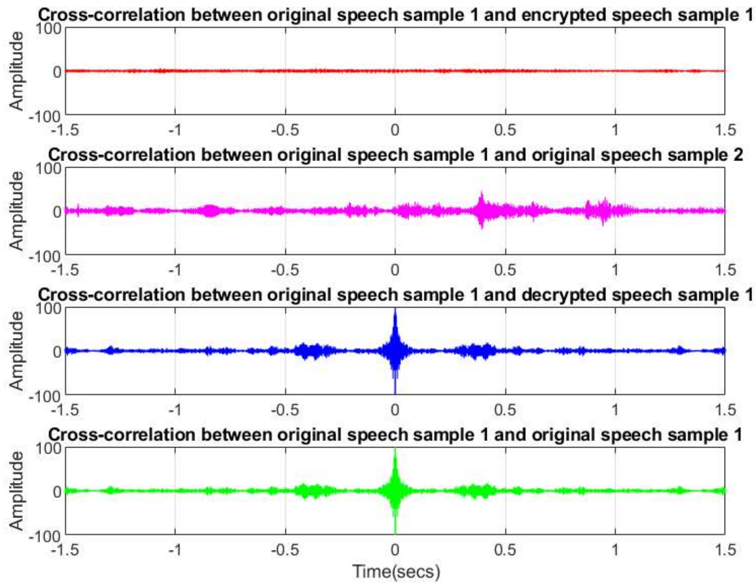
We can cross-correlate encrypted speech signal to original speech signal with the *xcorr* function (inbuilt MATLAB function) to determine if there is a match.

We can also find out the correlation coefficient between original signal and encrypted signal, and coefficient between original signal and decrypted signal.

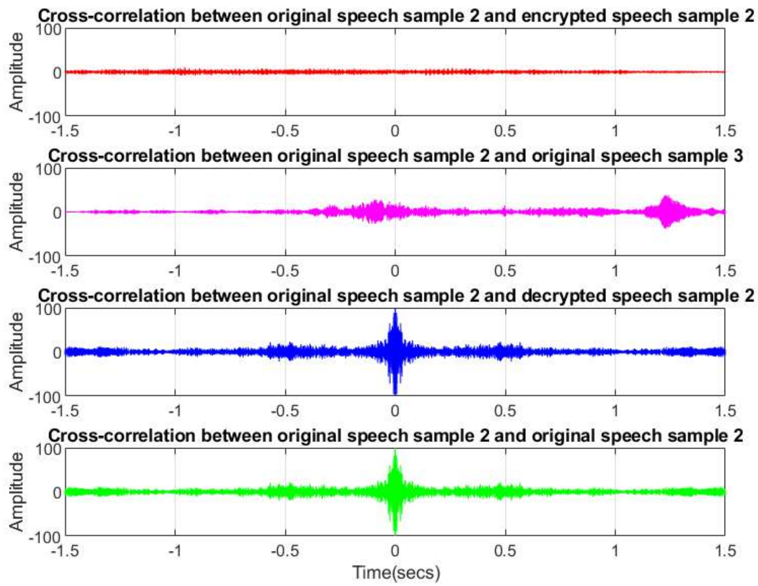
```
orig_decr_coeffs = [];
orig_encr_coeffs = [];

for m = 1:length(waves)
    [C1,lag1] = xcorr(original_speech_waveforms{m},encrypted_speech_waveforms{m});
    [C3, lag3] = xcorr(original_speech_waveforms{m},decrypted_speech_waveforms{m});
    [C4, lag4] = xcorr(original_speech_waveforms{m},original_speech_waveforms{m});

    if m == length(waves)  % if final wavefile is reached, it's orig speech signal will be correlated with that of the 1st wavefile
        n = 1;
        [C2,lag2] = xcorr(original_speech_waveforms{length(waves)},original_speech_waveforms{n});
    else % orig speech signal of current wavefile is correlated with the orig speech signal of next wavefile
        n = m+1;
        [C2,lag2] = xcorr(original_speech_waveforms{m},original_speech_waveforms{n});
    end

    % getting correlation coefficients
    R_1 = corrcoef(original_speech_waveforms{m}, decrypted_speech_waveforms{m});
    coeff_1 = R_1(2,1);
    orig_decr_coeffs = [orig_decr_coeffs coeff_1];

    R_2 = corrcoef(original_speech_waveforms{m}, encrypted_speech_waveforms{m});
    coeff_2 = R_2(2,1);
    orig_encr_coeffs = [orig_encr_coeffs coeff_2];

    fprintf('***** Audio Sample no. : %d *****', m);
    fprintf('correlation coefficient between decrypted and original speech is %f',coeff_1);
    fprintf('correlation coefficient between encrypted and original speech is %f',coeff_2);
    fprintf('Cross correlation plots')

    figure
    ax(1) = subplot(4,1,1);
    plot(lag1/fs,C1,'r')
    ylabel('Amplitude')
    grid on
    title(['Cross-correlation between original speech sample ' num2str(m) ' and encrypted speech sample ' num2str(m)])

    ax(2) = subplot(4,1,2);
    plot(lag2/fs,C2,'m')
    ylabel('Amplitude')
    grid on
    title(['Cross-correlation between original speech sample ' num2str(m) ' and original speech sample ' num2str(n)])

    ax(3) = subplot(4,1,3);
    plot(lag3/fs,C3,'b')
    ylabel('Amplitude')
    grid on
    title(['Cross-correlation between original speech sample ' num2str(m) ' and decrypted speech sample ' num2str(m)])

    ax(4) = subplot(4,1,4);
    plot(lag4/fs,C4,'g')
    ylabel('Amplitude')
```

```
    grid on
    title(['Cross-correlation between original speech sample ' num2str(m) ' and original speech sample ' num2str(m)])

    xlabel('Time(secs)')
    axis(ax(1:4),[-1.5 1.5 -100 100 ])
end
```

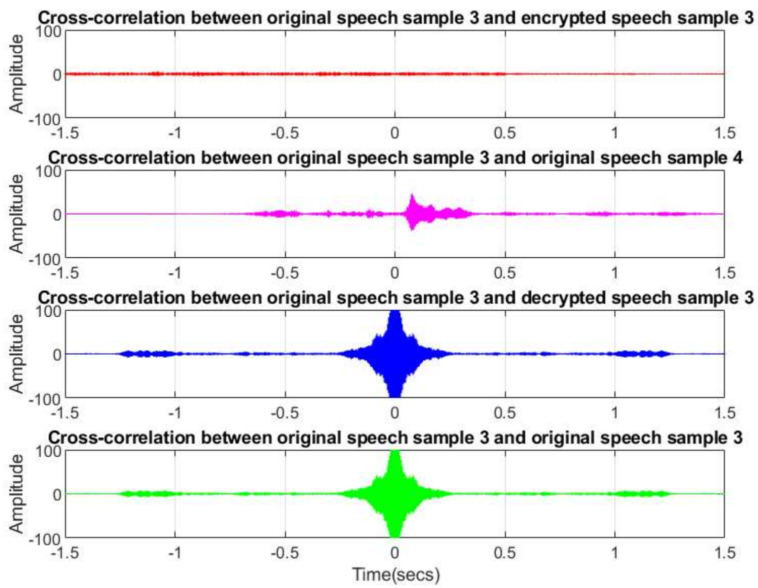
```
***** Audio Sample no. : 1 *****
correlation coefficient between decrypted and original speech is 0.999997
correlation coefficient between encrypted and original speech is -0.001623
Cross correlation plots
```




```
***** Audio Sample no. : 2 *****
correlation coefficient between decrypted and original speech is 0.999996
correlation coefficient between encrypted and original speech is -0.002530
Cross correlation plots
```

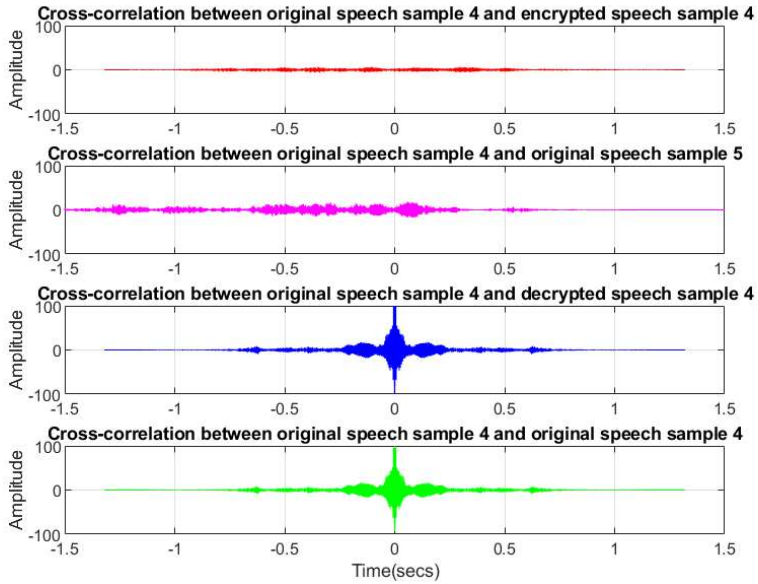



```
***** Audio Sample no. : 3 *****
correlation coefficient between decrypted and original speech is 0.999996
correlation coefficient between encrypted and original speech is -0.003815
Cross correlation plots
```

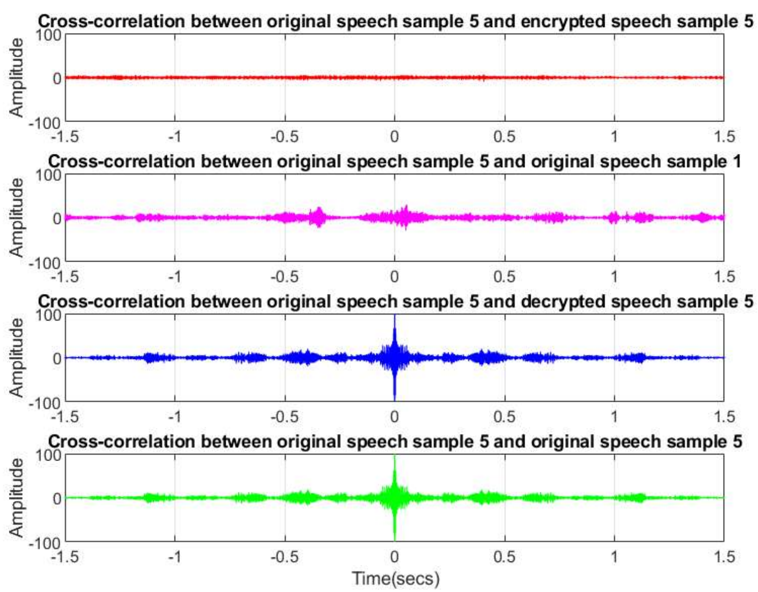



```
***** Audio Sample no. : 4 *****
correlation coefficient between decrypted and original speech is 0.999997
```

```
correlation coefficient between encrypted and original speech is -0.017292
Cross correlation plots
```



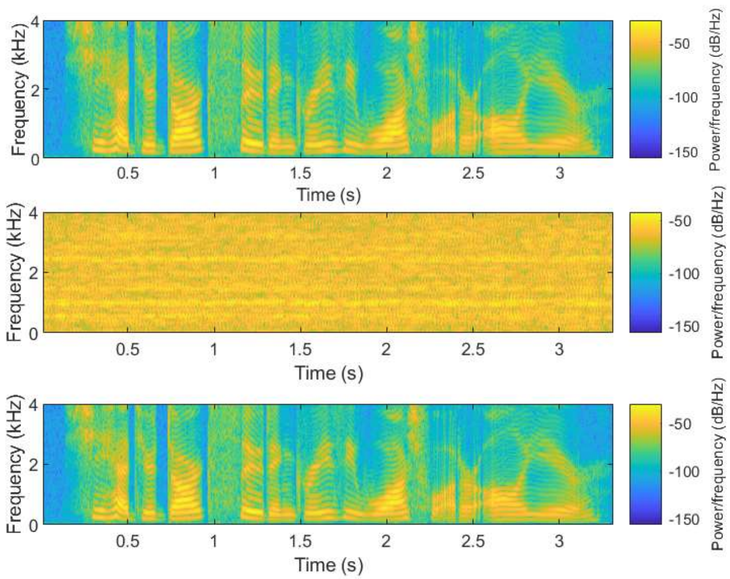

```
***** Audio Sample no. : 5 *****
correlation coefficient between decrypted and original speech is 0.999997
correlation coefficient between encrypted and original speech is 0.004653
Cross correlation plots
```



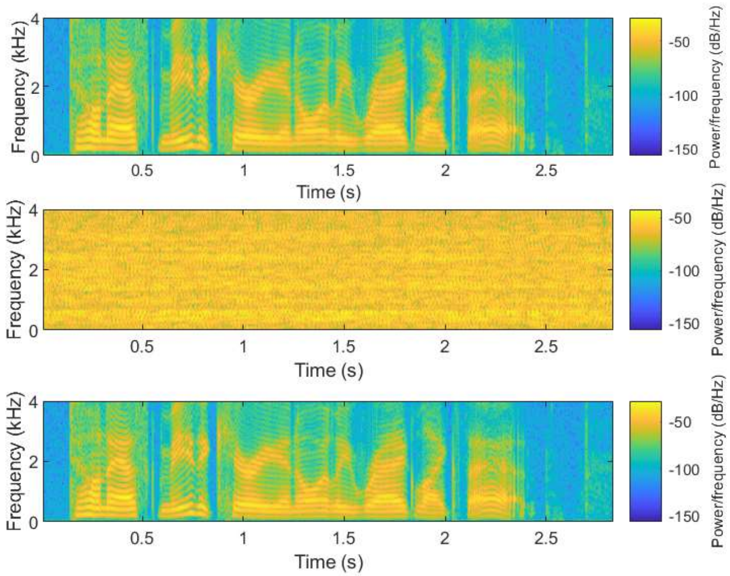
2.Using Spectrograms:

```
for jj = 1:length(waves)
    figure
    fprintf('Spectrograms of Original Speech signal, Encrypted Speech signal and Decrypted speech signal for audio sample %d', jj)
    subplot(311)
    spectrogram(original_speech_waveforms{jj}, 128, 125, 512, fs, 'yaxis');
    subplot(312)
    spectrogram(encrypted_speech_waveforms{jj}, 128, 125, 512, fs, 'yaxis');
    subplot(313)
    spectrogram(decrypted_speech_waveforms{jj}, 128, 125, 512, fs, 'yaxis');
end
```

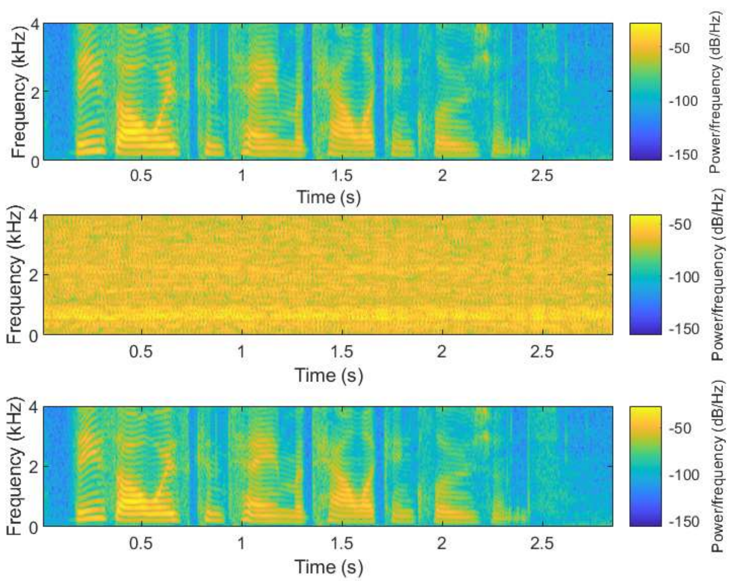
```
Spectrograms of Original Speech signal, Encrypted Speech signal and Decrypted speech signal for audio sample 1
```



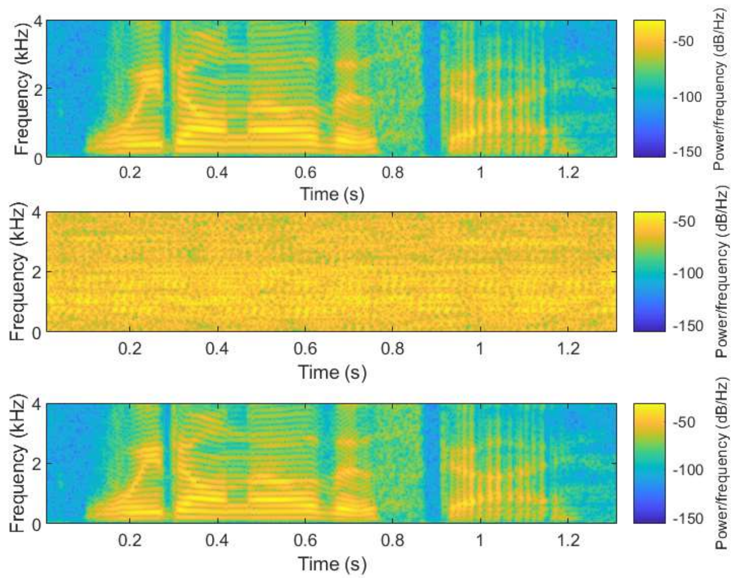

```
Spectrograms of Original Speech signal, Encrypted Speech signal and Decrypted speech signal for audio sample 2
```



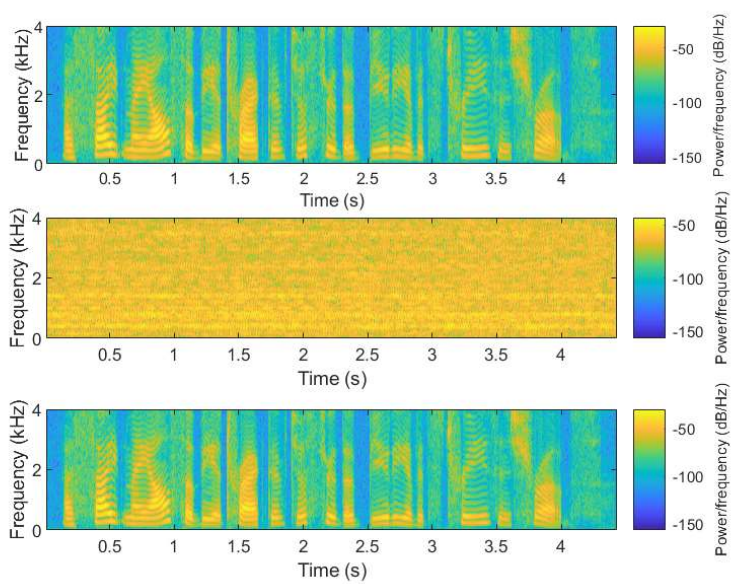
Spectrograms of Original Speech signal, Encrypted Speech signal and Decrypted speech signal for audio sample 3



Spectrograms of Original Speech signal, Encrypted Speech signal and Decrypted speech signal for audio sample 4

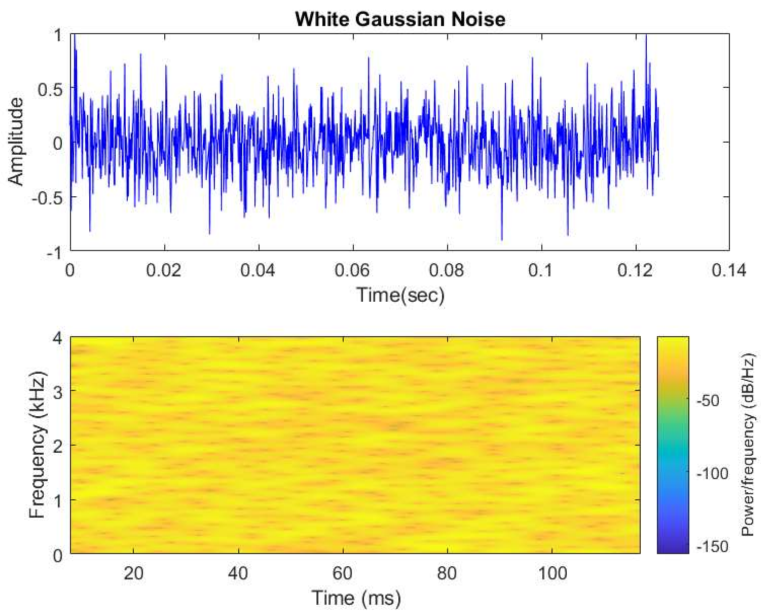


Spectrograms of Original Speech signal, Encrypted Speech signal and Decrypted speech signal for audio sample 5




```
% Comparing the spectrogram of the encrypted speech signal to that of the
% spectrogram of white gaussian noise
white_noise = wgn(1000,1,20);
figure
subplot(211)
```

```
plot((0:length(white_noise)-1)/fs,white_noise./max(white_noise), 'b'); xlabel('Time(sec)'); ylabel('Amplitude'); title('White Gaussian Noise')
subplot(212)
spectrogram(white_noise, 128, 125, 512, fs, 'yaxis');
```



3. Using SNR and PSNR values

SNR measures the noise content in the encrypted data signal. The proposed system has better negative SNR and so it is stronger against attacks.

$$SNR = 10 * \log_{10} \frac{\sum_{i=1}^{N_s} x_i^2}{\sum_{i=1}^{N_s} (x_i - y_i)^2}$$

(PSNR) is an expression for the ratio between the **maximum** possible **value** (power) of a signal and the power of distorting noise that affects the quality of its representation.

$$PSNR = 10 * log_{10} \left(\frac{MAX^2}{MSE} \right) \qquad MSE = \frac{1}{n} \sum_{i=1}^{n} (X[i] - Y[i])^2$$

Lower values of PSNR is desired for encrypted audio files as it refers to high level of noise in the encrypted audio files and so strong resistance against attacks.

```
SNR_vals = [];
PSNR_vals = [];
for ii = 1:length(waves)
    fprintf('SNR, PSNR values for audio sample %d', ii)

    xi = original_speech_waveforms{ii};
    yi = encrypted_speech_waveforms{ii};

    num = sum(xi.^2);
    den = sum((xi - yi).^2);

    snr = 10*log10(num/den)
```

```
    % calculating psnr
    p_snr = psnr(xi,yi)

    SNR_vals = [SNR_vals snr];
    PSNR_vals = [PSNR_vals p_snr];

end
```

```
SNR, PSNR values for audio sample 1
snr = -1.5602
p_snr = 16.7947
SNR, PSNR values for audio sample 2
snr = -1.5728
p_snr = 15.4915
SNR, PSNR values for audio sample 3
snr = -1.6610
p_snr = 17.4451
SNR, PSNR values for audio sample 4
snr = -1.6235
p_snr = 15.3172
SNR, PSNR values for audio sample 5
snr = -1.5093
p_snr = 18.6779
```

4. Percentage Residual Deviation Analysis:

Percentage Residual Deviation is a statistical tool to measure the variation of the encrypted speech signal from original signal. It can be seen that the encrypted signal is highly deviated from its original signal.

$$\emptyset = 100 \times \sqrt{\frac{\sum_{i=1}^{n}(x_i - y_i)^2}{\sum_{i=1}^{n} x_i^2}}.$$

```
PRD_vals = [];
for ii = 1:length(waves)
    fprintf('PRD values for audio sample %d', ii)
    xi = original_speech_waveforms{ii};
    yi = encrypted_speech_waveforms{ii};
    num = sum((xi - yi).^2);
    den = sum(xi.^2);
    PRD = 100*((num/den)^0.5)
    PRD_vals = [PRD_vals PRD];
end
```

```
PRD values for audio sample 1
PRD = 119.6772
PRD values for audio sample 2
PRD = 119.8507
PRD values for audio sample 3
PRD = 121.0739
PRD values for audio sample 4
```

PRD = 120.5516

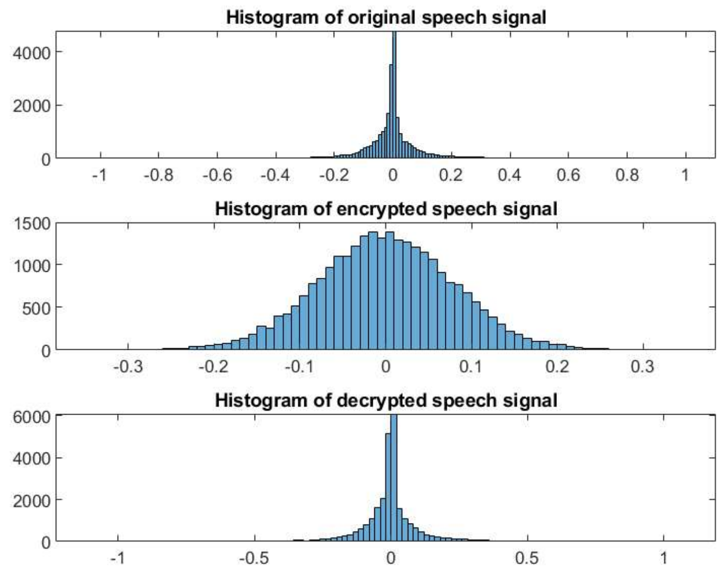
PRD values for audio sample 5

PRD = 118.9781

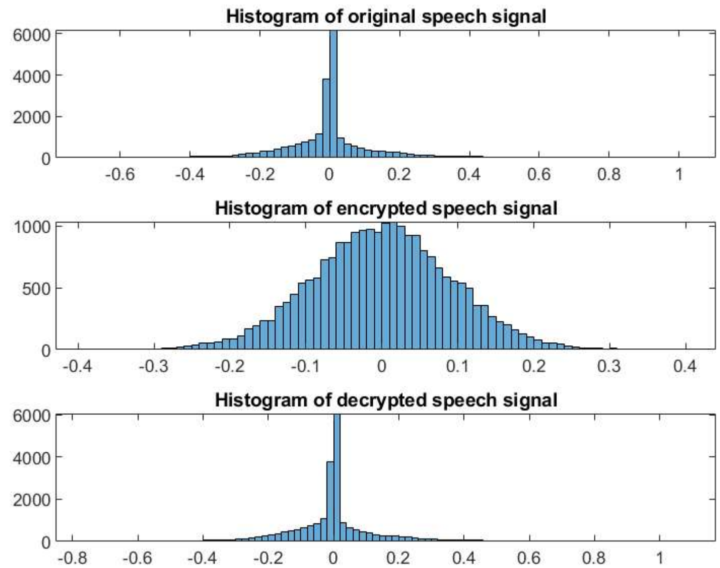
5. HISTOGRAM ANALYSIS:

```
for ii = 1:L
    fprintf('***** Audio sample: %d *****', ii)
    figure
    subplot(311); histogram(original_speech_waveforms{ii}); title('Histogram of original speech signal')
    subplot(312); histogram(encrypted_speech_waveforms{ii}); title('Histogram of encrypted speech signal')
    subplot(313); histogram(decrypted_speech_waveforms{ii}); title('Histogram of decrypted speech signal')
end
```

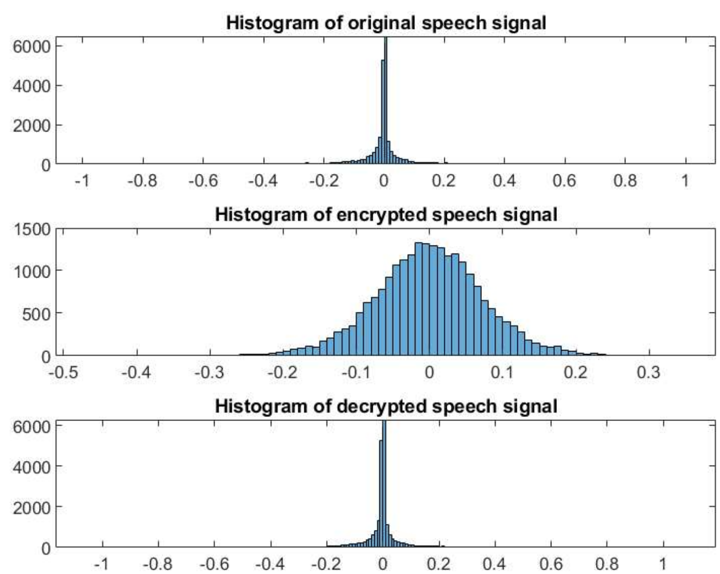
```
***** Audio sample: 1 *****
```



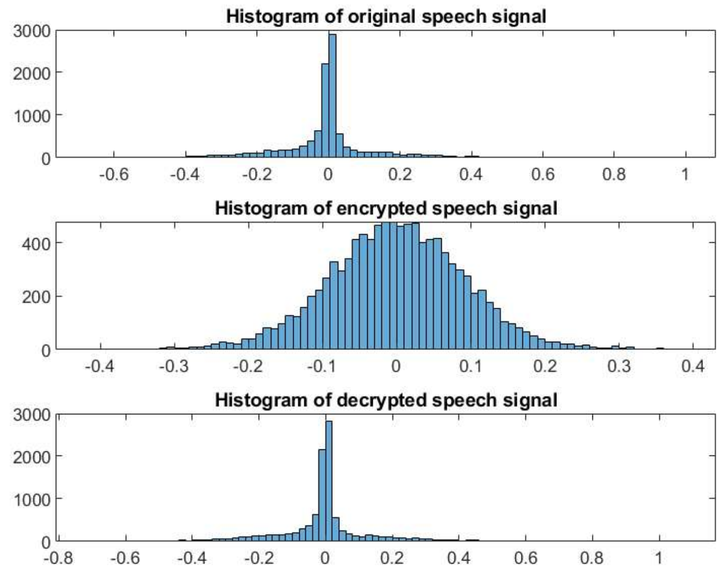

```
***** Audio sample: 2 *****
```



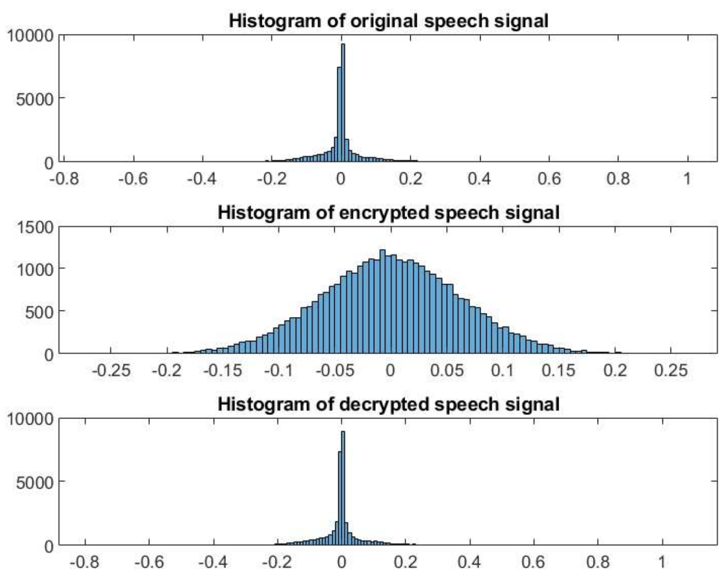

```
***** Audio sample: 3 *****
```




```
***** Audio sample: 4 *****
```



***** Audio sample: 5 *****



Comparison with other encryption systems

Correlation

TABLE II. CORRELATION COEFFICIENT OF ORIGINAL AND SCRAMBLED SAMPLES

Speech sample	Original signal	Encrypted signal
Digits	0.982	-0.016
Sentences	0.986	-0.022
Conversation	0.974	-0.028

Reference: Secure Speech Communication Algorithm via DCT and TD-ERCS Chaotic Map, Zeeshan Habib, Jan Sher Khan, Jawad Ahmad, Muazzam A. Khan, Fadia Ali Khan

TABLE III. EXISTING METHOD[1] (CORRELATION BETWEEN ORIGINAL & ENCRYPTED SIGNAL)

#	Sample Files	Correlation
1	A	0.000569
2	B	0.000819
3	C	0.000456

Reference: Farsana F J , Gopakumar K,A Novel Approach for Speech Encryption: Zaslavsky Map as Pseudo Random Number Generator

TABLE IV. EXISTING METHOD [2] (CORRELATION BETWEEN ORIGINAL & ENCRYPTED SIGNAL)

#	Sample Files	Correlation
1	Audio1.wav	0.0233
2	Audio2.wav	0.0384
3	Audio3.wav	0.0157

Reference: Sathiyamurthi and Ramakrishnan, Speech encryption using chaotic shift keying for secured speech communication

#	Sample Files	Correlation
1	TIMIT Corpus	-0.0015
2	OpenSLR	0.00011469
3	LDC-IL	-0.00020515

Reference: Voice Signal Encryption Scheme Using Transformation and Embedding Techniques for Enhanced Security, PL. Chithra and Aparna R.

Table 5 Correlation Analysis

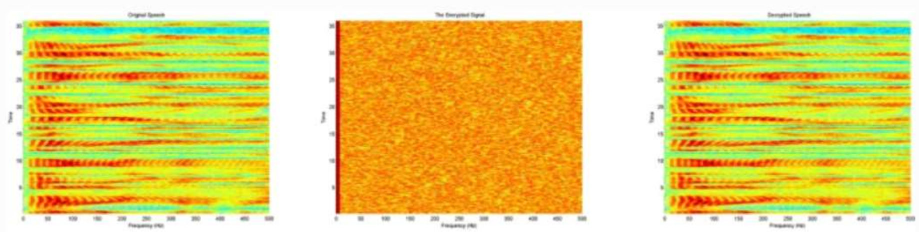
From: Speech encryption algorithm using FFT and 3D-Lorenz–logistic chaotic map

S.No	Speech File	Correlation Coefficient Original Vs Encrypted	Correlation Coefficient Original Vs Decrypted
1	Speech1.wav	0.0386	0.9958
2	Speech2.wav	−0.0974	−0.9925
3	Speech3.wav	0.0312	0.9917
4	Speech4.wav	−0.0643	−0.9896
5	Speech5.wav	0.0514	0.9899
6	Speech6.wav	0.0458	0.9927

Reference: Speech encryption algorithm using FFT and 3D-Lorenz–logistic chaotic map

Spectrogram analysis:

Fig. 4



a Spectrogram of Original Speech

b Spectrogram of Encrypted Speech

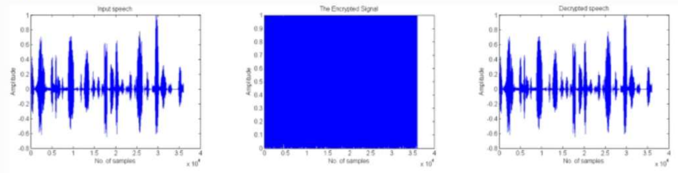
c Spectrogram of Decrypted Speech

Spectrogram of original, encrypted and decrypted speech

Reference: Speech encryption algorithm using FFT and 3D-Lorenz–logistic chaotic map

Histogram Analysis:

Fig. 3



a Original Speech b Encrypted Speech c Decrypted Speech

d Histogram of Original Speech e Histogram of Encrypted Speech f Histogram of Decrypted Speech

Plot and histogram of original, encrypted and decrypted speech

Reference: Speech encryption algorithm using FFT and 3D-Lorenz–logistic chaotic map

SNR Analysis

TABLE 8. SNR, PSNR and correlation coefficient between the original and encrypted audio files.

Audio file	Size	Duration (Sec.)	Correlation coefficient	SNR	PSNR
Audio-1	156 KB	1.8	0.00016	-28.53	4.25
Audio-2	304 KB	1.76	-0.00028	-28.14	4.31
Audio-3	1.84MB	11	0.00052	-37.08	4.35
Audio-4	1.02 MB	33.52	0.00026	-32.96	4.12
Audio-5	3.13 MB	18.6	0.00041	38.66	4.30
Audio-6	543 KB	3.15	0.00040	-29.96	4.34
Audio-7	3.19MB	19	0.000053	-38.02	4.37
Audio-8	984 KB	5.7	0.00057	-32.95	4.33

Sample files	SNR	Correlation	PRD(Ø)
A. Male voice	−12.45 dB	0.00669	0.521×10^5
B. Female voice	−13.89 dB	0.00918	0.689×10^6
C. Male voice	−21.89 dB	0.00693	0.723×10^5
D. Female voice	−14.32 dB	0.00229	0.214×10^5
E. Male voice	−22.89 dB	0.00527	0.934×10^5
F. Female voice	−19.45 dB	0.00358	0.394×10^6
G. Male voice	−11.76 dB	0.00992	0.861×10^5
H. Female voice	−23.23 dB	0.00136	0.231×10^6

```
% correlation coeffs between original speech and decrypted original speech
orig_decr_coeffs
```

```
orig_decr_coeffs = 1×5
    1.0000    1.0000    1.0000    1.0000    1.0000
```

```
% correlation coeffs between original speech and encrypted original speech
orig_encr_coeffs
```

```
orig_encr_coeffs = 1×5
   -0.0016   -0.0025   -0.0038   -0.0173    0.0047
```

```
SNR_vals
```

```
SNR_vals = 1×5
   -1.5602   -1.5728   -1.6610   -1.6235   -1.5093
```

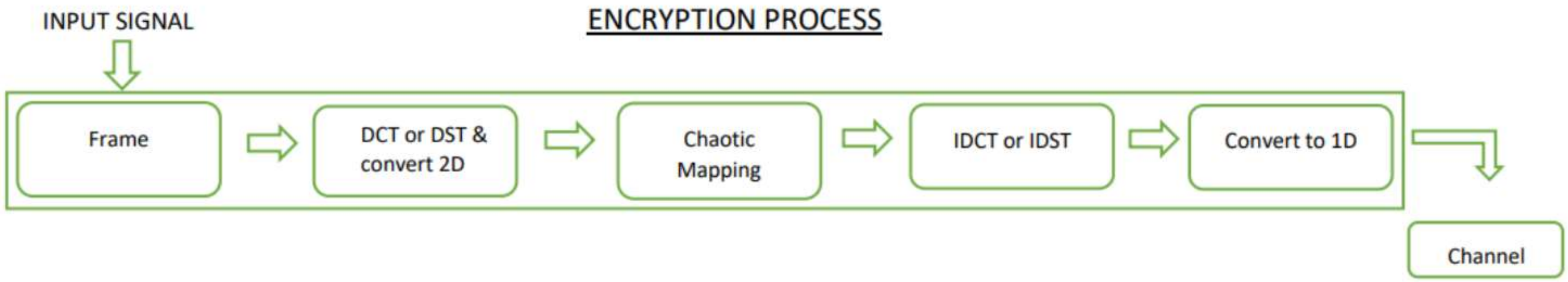
```
PSNR_vals
```

```
PSNR_vals = 1×5
   16.7947   15.4915   17.4451   15.3172   18.6779
```

```
PRD_vals
```

```
PRD_vals = 1×5
  119.6772  119.8507  121.0739  120.5516  118.9781
```

FUNCTION DEFINITIONS



ENCRYPTION:

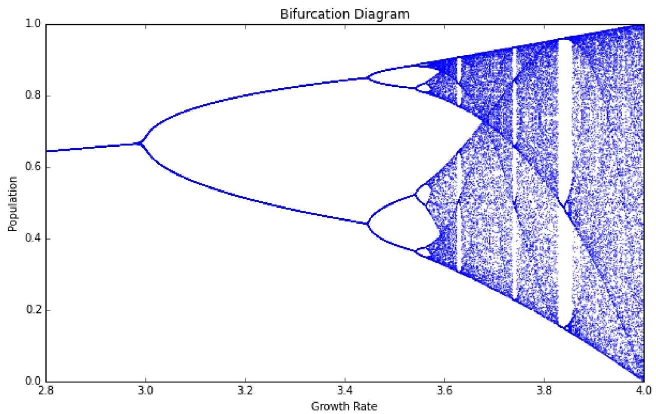
```
function [Encrypted_speech_1D, Chaotic_map] = encryption(sample_sound, fs)
```

FRAME AND CONVERT TO 2D AFTER APPLYING DCT

```
    frame_size = 0.020;
    frame_shift = 0.010; % COLA(M/2) for reconstruction (when hamming window is applied)
    frame_length = frame_size*fs; % no. of samples in one frame (here it is 160 samples for 20 ms frame size, fs = 8000 Hz)

    % getting no. of frames
    no_of_frames = length(0:frame_shift:length(sample_sound)/fs-frame_size);
    n = 0;

    x_frame1 = [];
    x_frame2 = [];

    signal = [];
    dct_signal = [];

    m = 1;

    % Framing and applying DCT to the frames and summing all these DCTs
    for i = 1:no_of_frames
        x = sample_sound((1+round((n*frame_shift)*fs)):(round((n*frame_shift + frame_size)*fs))); % subsignal corresponding to a frame

        % applying hamming window to this subsignal
        win = dsp.Window("Hamming");
        x_win = win(x);

        % summing up all dcts of frames and storing it in dct_signal
        if i == 1
            signal = [signal x_win];
            dct_signal = [dct_signal dct(x_win)];
        else
```

```
        x_frame1 = [signal; zeros(frame_shift*fs, 1)]; % zero padding at end by 80 samples
        x_frame2 = [zeros(round(m*frame_shift*fs), 1); x_win]; % zero padding at start by m*80 samples (m is an integer)
        % We should add 2nd half elements of x_frame1 with the 1st half
        % elements of x_frame2 (after applying dct to these elements)
        % overlap is M/2
        m = m + 1;
        signal = x_frame1 + x_frame2;
        dct_signal = dct(x_frame1) + dct(x_frame2); % sum of all dcts of windowed subsignals x_win i.e. Summation of X_m(w) over all m which is same as X(w)
    end

    n = n+1;
end

% converting to 2-D for applying chaotic mapping
dct_frames_matrix = reshape(dct_signal, frame_length, (length(dct_signal)/frame_length));
dct_frames_matrix = dct_frames_matrix';

Transformed_frames = dct_frames_matrix;
```

CHAOTIC MAPPING




```
x1 = 0.1;
r = 4;

for k = 1: size(dct_frames_matrix, 1)
    for j = 1:size(dct_frames_matrix, 2)
        Chaotic_map(k, j) = r*x1*(1-x1);
        x1 = Chaotic_map(k, j);
    end
end

Randomized_matrix = Transformed_frames.*Chaotic_map;
```

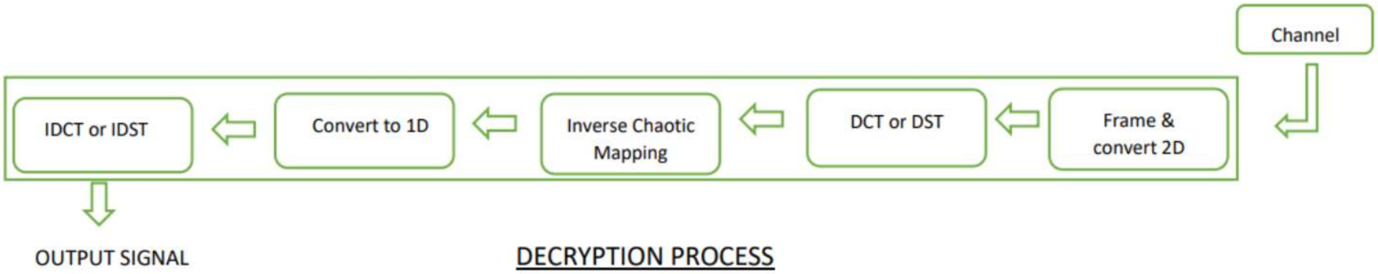
IDCT

```
Encrypted_speech_2D = idct(Randomized_matrix);
```

CONVERT TO 1-D

```
    Encrypted_speech_1D = reshape(Encrypted_speech_2D',[],1); % convert matrix to column vector i.e. to 1-D
end
```

DECRYPTION



DECRYPTION PROCESS

```
function [Original_1D_matrix] = decryption(Encrypted_speech_1D, Chaotic_map, fs)
```

FRAME AND CONVERT TO 2D

```
    frame_size = 0.020;
    frame_length = frame_size*fs;
    Received_speech_2D = reshape(Encrypted_speech_1D, frame_length, (length(Encrypted_speech_1D)/frame_length));
    Received_speech_2D = Received_speech_2D';
```

DCT

```
    Transformed_Encrypted_speech_2D = dct(Received_speech_2D);
```

INVERSE CHAOTIC MAPPING

```
    Unrandomized_matrix = Transformed_Encrypted_speech_2D./Chaotic_map;
```

CONVERT TO 1-D

```
    Unrandomized_1D_matrix = reshape(Unrandomized_matrix',[],1); % same as fft signal
```

IDCT

```
    Original_1D_matrix = idct(Unrandomized_1D_matrix);
end
```