Encryption and Decryption of Audio Signals

Subject: Signals and Systems

Instructor: Prof. Ashok Ranade

Group no.:16

Group Members:

Name	Roll no.	Email
Rushil Sojitra	AU1841096	rushil.s1@ahduni.edu.in
Nihir Dhamecha	AU1841117	nihir.d@ahduni.edu.in
Preet Rajpara	AU1841060	preet.r@ahduni.edu.in
Jimil Desai	AU1841147	jimil.d@ahduni.edu.in

Group Photo:

Introduction:

This present work puts forth a novel method to encrypt '*.wav' audio files, into image files with formats such as PNG, TIF, and JPEG. The sound file is fetched and the values corresponding to the sample range are put in a column matrix which is then reshaped into a two-dimensional matrix having "double" as data-type. This matrix is defined in the class 'double' and is put in a grayscale image file. After encryption, the data is retrieved from the image file and compared with the original wave file to show the variation in encrypted and decrypted data. This methodology can not only be used as stenographic means but possibly as a technique for data compression. The illustrated method for data encryption of sensitive user information can be used as a viable method to further secure cloud computing transactions.

Theories Used:

Here, in this method Linear Pulse Code Modulation (LPCM) is used which enables us to represent sampled analog data in digital form. Also, it allows us to store the samples in double type format i.e. from -1 to 1 in a column matrix which can then be used to convert the data into an image and the image can then be transferred over long distances and then can be decrypted with our decryption tool which uses demodulation to convert the signal back to analog format using the standard 44.1 kHz frequency.

Working/Algorithm:

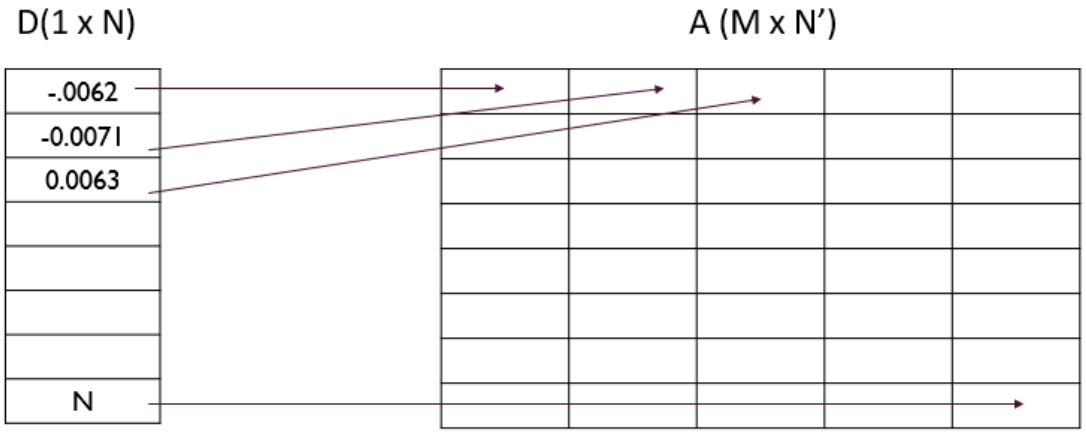
Step 1: Obtaining data from a wave file in the column matrix

First of all, we will fetch the wave signal, then the wave file will be demodulated to get the samples and sample length. We will be considering the full-wave i.e. positive(till 1) as well as negative values(till -1) of the wave and all the samples will be stored in a column matrix D. Waveform Audio File Format (WAVE) is an

application of RIFF or Resource Interchange File Format which stores audio bitstreams in “chunks”.We use WAVE format because it is a lossless format and it is widely used for professional recordings.

Step 2: Converting the column matrix into MXN’ MATRIX:

Variable D is a column matrix with a “double” type and intensity within -1 and 1. So to convert variable D in an image format, we have to transform D into an M x N’ matrix. Matrix A is an M X N’ matrix having a “double” data type wherein each element of the matrix denotes a pixel within an intensity of -1 and 1.



Step 3:Converting Matrix into an Image file:

We can convert matrix A into JPEG, PNG and TIFF formats. We will be using a TIF format because it is a lossless compression format. We will save the image in grayscale instead of true color(RGB) as it preserves the minimum and maximum values of the matrix rather than preserving values between 0 and 1 only. We will be saving variable D for reference. At the end of this, we will be getting an encrypted image of the Audio signal.

Step 4: Decryption of the Encrypted method:

Decryption is just the opposite of encryption with minor variations. When the image is created during encryption it is basically an M x N matrix with a “double” data type, however on fetching the same image back we get an M x N matrix with datatype “uint8” i.e. unsigned integer of 8 bits. Thus, we need to convert all elements of the matrix obtained into double precision. Decryption can be basically understood as a data mining method to fetch, audit and understand the pattern of data stored in the encrypted file.

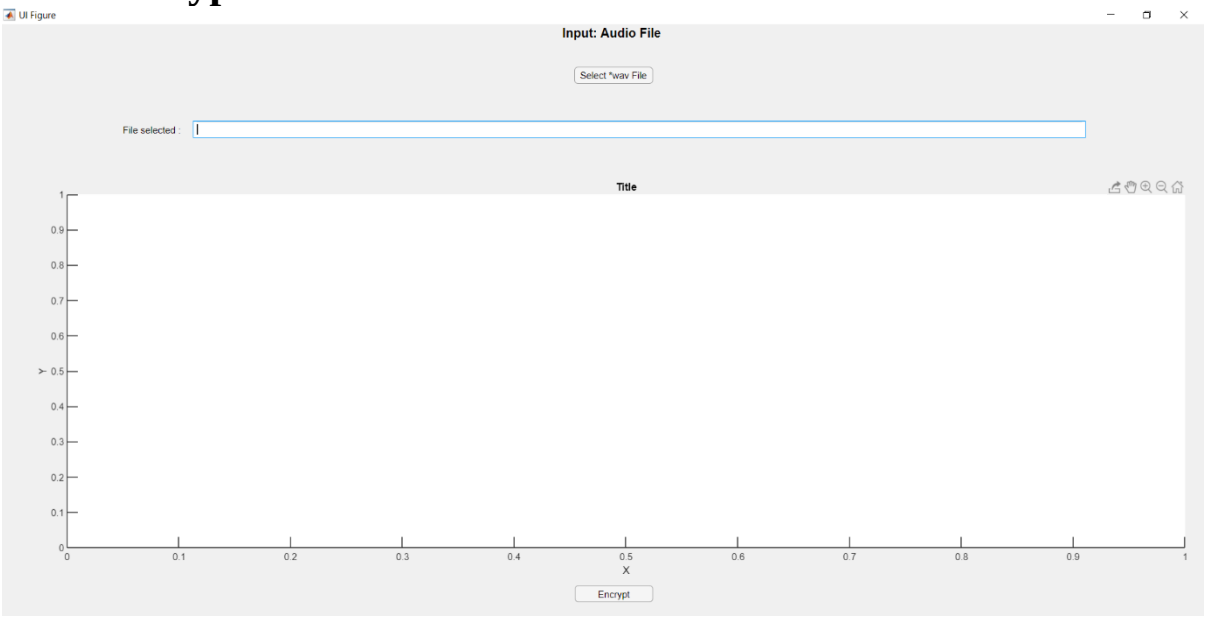
In a Nutshell:

Encryption:

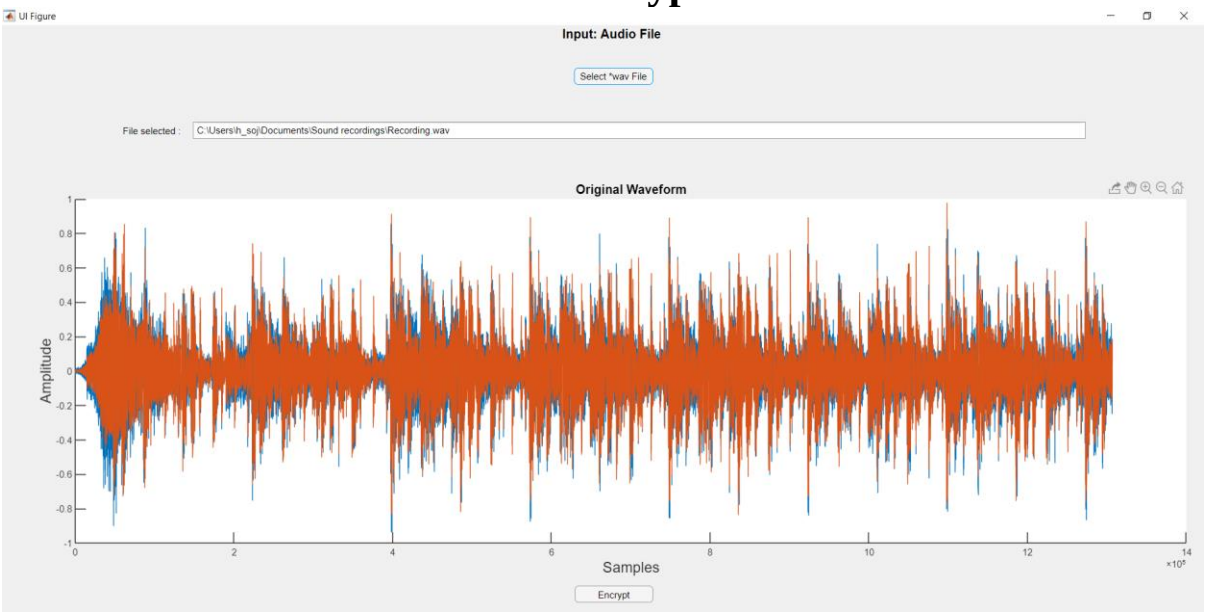
Decryption:

RESULTS:

1. Encryption Screen

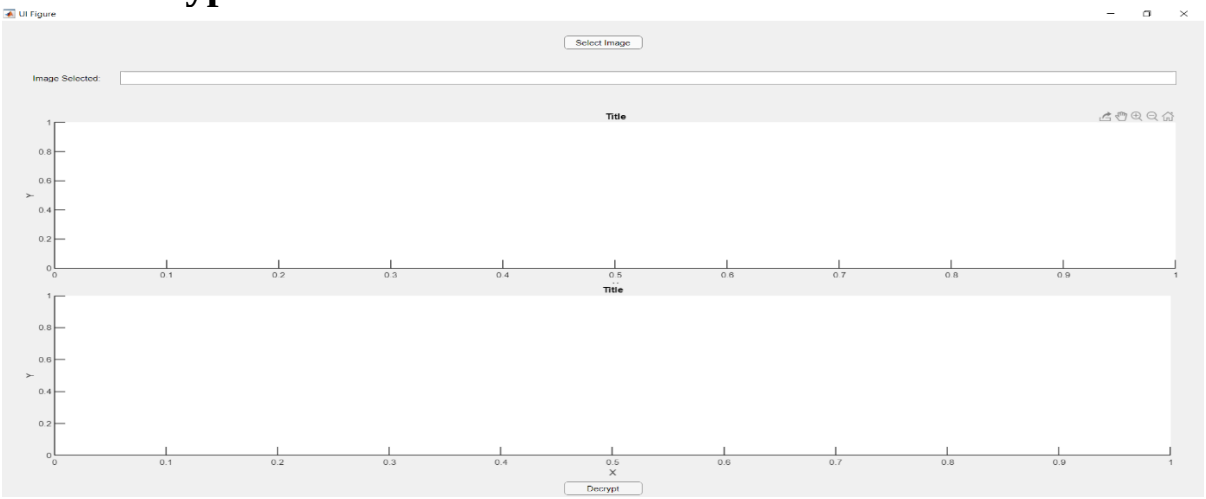


2. Selection of *.wav file for encryption

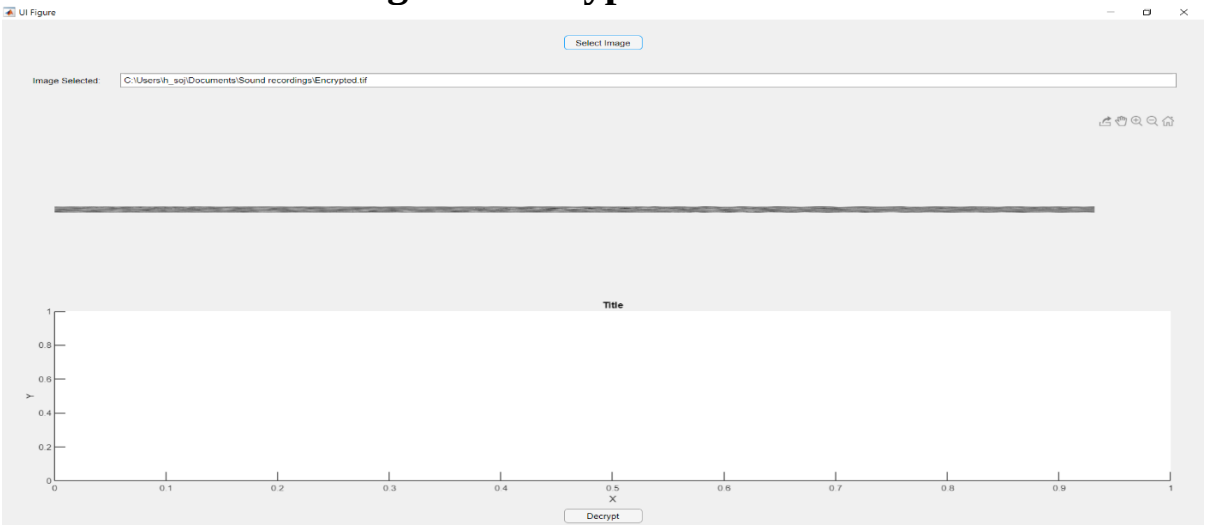


3. Image obtained after encryption

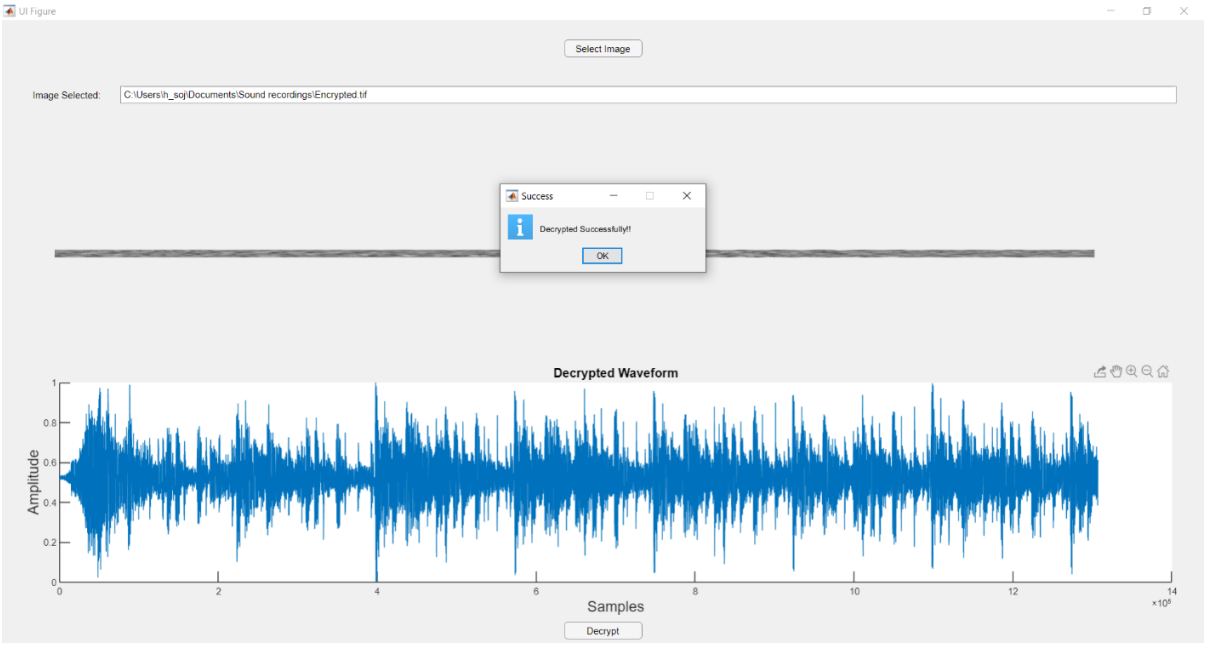
4. Decryption Screen



5. Selection of image for decryption



6. Final Waveform



Source Code:

Editor - E:\Signals and Systems\Project\Original_Waveform.m

ReadWAV.m | Original_Waveform.m | To_Image.m | TO_WAV.m

```
%This function is used to plot the original waveform of the recorded *.wav
%file that is to be encrypted
function y = Original_Waveform(path,file)
y = audioread(fullfile(path,file));
```

Editor - E:\Signals and Systems\Project\ReadWAV.m

ReadWAV.m | Original_Waveform.m | To_Image.m | TO_WAV.m

```
%This function is used to read the samples of the *.wav file that is to be
%encrypted.
function ReadWAV(path,file)
%The next command performs pulse code modulation which is used to represent
%sampled analog signals in digital form and stores the samples in the
%variable b and the sampling frequency in the variable Fs. The values of
%samples that are stored in b would be of type double i.e. from -1 to 1.
[b,Fs] = audioread(fullfile(path,file));
C=b;
%The number of samples is stored in the variable X with the following
%command.
X=length(b);
%Now, in the follwing command, a column vector D of size same as the number
%of samples in b, is created and its values are assigned as the samples.
D=C(1:X);
%The next command is a call to a function which will convert the vector to
%an image.
To_Image(X,D,path);
```

Editor - E:\Signals and Systems\Project\To_Image.m

ReadWAV.m | Original_Waveform.m | To_Image.m | TO_WAV.m

```
%This function performs encryption by converting the vectror received from
%ReadWAV function into a *.tif image file. Here, *.tif format is used
%beacuse the TIFF file format is a very good lossless compression algorithm
%for image.
function To_Image(X,D,path)
%First the column vector needs to be reshaped into a m x n matrix. Here, m
%is set to 100 and the number of columns i.e. n varies according to the
%number of samples in the *.wav file.
x=1;
y=round(X/100,0);
%This loop reshapes the column vector into the matrix and fills the matrix
%with all the values.
for i=1:100
    L=D(x:y);
    N=L';
    A(i,:)=N;
    x=x+round(X/100);
    y=y+round(X/100);
end
%Here, the matrix that is formed, is stored as a *.tif image file and also
%the image is in grayscale to preserve the double values of the samples
%that are received from the *.wav file.
    imwrite(mat2gray(A),fullfile(path,'Encrypted.tif'));
```

Editor - E:\Signals and Systems\Project\TO_WAV.m

ReadWAV.m | Original_Waveform.m | To_Image.m | TO_WAV.m

```
%This function is used to decrypt the image that we had formed in the
%To_Image function again into a *.wav file.
function TO_WAV(path,file)
%The following command will read the image from the file path specified and
%then it will store it into the variable aatif.
aatif=imread(fullfile(path,file));
%The image is then again converted into grayscale to obtain the double
%values.
mat2gray(aatif);
%The follwing command converts the image into a matrix of the type double
%and again stores in the varible aatif.
aatif=im2double(aatif);
TIF=[;];
%The next loop converts the matrix again into a column vector TIF by
%reshaping the matrix again into a column by vertical concatenation.
for i=1:100
    w=aatif(i,:);
    m=w';
    TIF=vertcat(TIF,m);
end
%After obtaining the column vector from the above loop we can again write
%it into a *.wav file with a sampling frequency of 44.1 KHz which is a
%standard sampling frequency. Thus, after the follwing command we will have
%successfully completed decryption and we will get the audio file almost
%similar to the original audio file. Actually, the audio file after
%decryption will also get compressed as the image is stored in TIFF format.
audiowrite(fullfile(path,'Recording2.wav'),TIF,44100);
```

Command Window

```
    FileselectedEditField  matlab.ui.control.EditField
    UIAxes                 matlab.ui.control.UIAxes
    EncryptButton          matlab.ui.control.Button
end

properties (Access = public)
    file % Description
    path
end

% Callbacks that handle component events
methods (Access = private)

    % Code that executes after component creation
    function startupFcn(app)
    drawnow;
        app.UIFigure.WindowState = 'maximized';

    end

    % Callback function
    function RecordAudioButtonPushed(app, event)

    end

    % Callback function: FileselectedEditField,
    % SelectwavFileButton
    function SelectwavFileButtonPushed(app, event)
        [app.file,app.path] = uigetfile('*.wav');
        drawnow;
        figure(app.UIFigure);
        app.FileselectedEditField.Value = fullfile(app.path,app.file);
        y = Original_Waveform(app.path,app.file);
        plot(app.UIAxes,y);
        app.UIAxes.Title.String = 'Original Waveform';
        app.UIAxes.Title.FontSize = 16;
        app.UIAxes.XLabel.String = 'Samples';
        app.UIAxes.XLabel.FontSize = 14;
        app.UIAxes.YLabel.String = 'Amplitude';
        app.UIAxes.YLabel.FontSize = 14;

    end

    % Callback function
    function FileselectedEditFieldValueChanged(app, event)

    end

    % Button pushed function: EncryptButton
    function EncryptButtonPushed(app, event)
        %file = app.SelectwavFileButton
        %path = get(handles.SelectwavFileButtonPushed,'path');
            ReadWAV(app.path,app.file);
            msgbox('Encrypted Successfully!!','Success','help');

    end
end
```

```
classdef app2 < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                    matlab.ui.Figure
        SelectImageButton           matlab.ui.control.Button
        ImageSelectedEditFieldLabel  matlab.ui.control.Label
        ImageSelectedEditField      matlab.ui.control.EditField
        UIAxes                      matlab.ui.control.UIAxes
        DecryptButton               matlab.ui.control.Button
    end


    properties (Access = public)
        file;
        path;% Description
    end


    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)
              drawnow;
            app.UIFigure.WindowState = 'maximized';

        end

        % Button pushed function: SelectImageButton
        function SelectImageButtonPushed(app, event)
            [app.file,app.path] = uigetfile('*.tif');
            drawnow;
            figure(app.UIFigure);
            app.ImageSelectedEditField.Value = fullfile(app.path,app.file);
            Image1 = imread(fullfile(app.path,app.file));
            imshow(Image1,'Parent',app.UIAxes);
            app.UIAxes.XLabel.String = '';
            app.UIAxes.Title.String = '';
            app.UIAxes.Title.String = '';
        end

        % Button pushed function: DecryptButton
        function DecryptButtonPushed(app, event)
            TO_WAV(app.path,app.file);
            msgbox('Decrypted Successfully!!','Success','help');
        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [100 100 640 480];
            app.UIFigure.Name = 'UI Figure';
```