

INSTITUTIONEN FÖR TEKNIK OCH
NATURVETENSKAP

LINKÖPINGS UNIVERSITET

TNG015: SIGNALER OCH SYSTEM, VT 2019

Ljudkryptering och analys med MATLAB

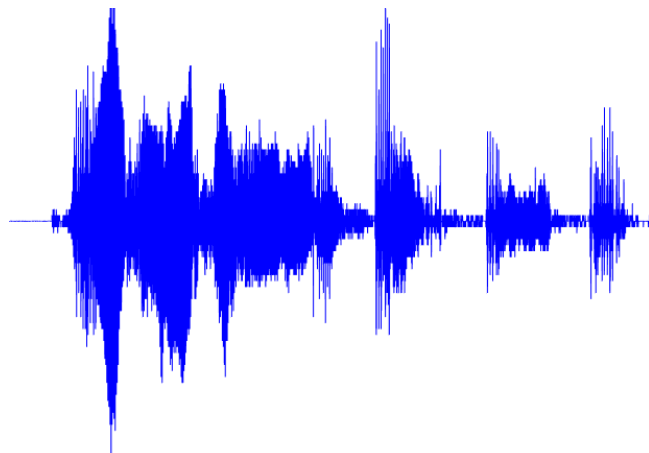
16 juni 2019

Författare

PHILIP NGO: *phing272@student.liu.se*

Examinator

OLE PEDERSEN: *ole.pedersen@liu.se*



1 Sammanfattning

I denna uppsats har ett ljud krypteringsprogram skapats i MATLAB. Krypteringsprogrammet delades in i tre delar: 1. Spela in ljudklippet, 2. Kryptera ljudklippet och 3. Dekryptera ljudklippet. Programmet skapades och testades genom att låta en användare starta programmet i MATLAB och spela in ett ljudklipp som därefter analyserades. Ljudklippet analyserades både i tidsdomänen och frekvensdomänen så att krypteringen av ljudfilen kunde appliceras på lämpligt sätt. Detta gjordes dels med hjälp av de inbyggda funktionerna i MATLAB och dels egna funktioner. Krypteringen gjordes genom att lägga på störningar i ljudklippet för i form av brus och få ljudklippet oanvändbart.

Genom att spara ner vilka störningar som applicerades i form av två vektorer kunde dessa två vektorer, som i denna rapporten kallas för *keys* eller *nycklar* (brist på en bättre term), användas för att återställa den krypterade ljudfilen. Det går i princip inte att rädda ljudfilen utan *nycklarna*. Varje *nyckel* är en vektor som genereras slumpmässigt med avseende på ljudvågornas amplitud. Detta är på grund av att den krypterade ljudklippet ska ha ett amplitudsintervall mellan $[-1, 1]$ eftersom MATLAB skriver endast ner de värdena ljudfilen när det sparas. Användaren kan i ljud-krypteringsprogrammet välja andra alternativ, utöver krypteringen, såsom att bestämma tiden på inspelningen eller att visa data för både den krypterade ljudklippet och den originella. När krypteringen är klar så sparas den krypterade ljudfilen som en .wav fil och "nycklarna" sparas som .txt filer.

Innehållsförteckning

1	Sammanfattning	1
2	Inledning	3
2.1	Bakgrund	3
2.2	Syfte	3
2.3	Metod och källor	3
2.4	Struktur	3
3	Genomförande / Redogörelse för arbetet	3
3.1	Del 1: Spela in ljud i MATLAB	3
3.2	Del 2: Kryptera ljudklippet	4
3.3	Del 3: Dekryptera ljudklippet	5
4	Resultat och analys	6
4.1	Resultat: Exempel på testkörning av programmet	6
4.2	Analys och slutsatser	8
A	Körning av programmet	10
B	Fullständig MATLAB kod	14

2 Inledning

2.1 Bakgrund

Det finns idag många ljudmanipulerings program som exempelvis manipulerar ljudklippet så att det låter ljusare, mörkare eller även robotliknande. Det finns dock inte lika många ljud krypteringsprogram enligt personliga observationer. Detta gav upphov till frågan om det helt enkelt var för svårt att skapa en krypterad ljudfil eller om efterfrågan på ett sådant program var låg.

2.2 Syfte

Syftet med rapporten är att undersöka om ett sådant program kan skapas i MATLAB med endast grundläggande kunskaper inom MATLAB, samt svara på frågan "är det för svårt att skapa en krypterad ljudfil eller om efterfrågan på ett sådant program är låg?".

Hypotesen är att det inte ska vara så svårt att kryptera en ljudfil eftersom en ljudfil kan manipuleras på många olika sätt i MATLAB och att anledningen till att det finns få krypteringsprogram är endast på grund av låg efterfråga.

2.3 Metod och källor

Metoden är att skriva koden på egen hand med undantaget att söka upp färdiga funktioner på MATLAB:s egna hemsida [1].

2.4 Struktur

Arbetsstrukturen är att börja koda huvuddelar av programmet separat och försöka få varje del att fungera och sedan sätta ihop det till ett färdigt program som klarar av att kryptera och dekryptera ett ljudklipp.

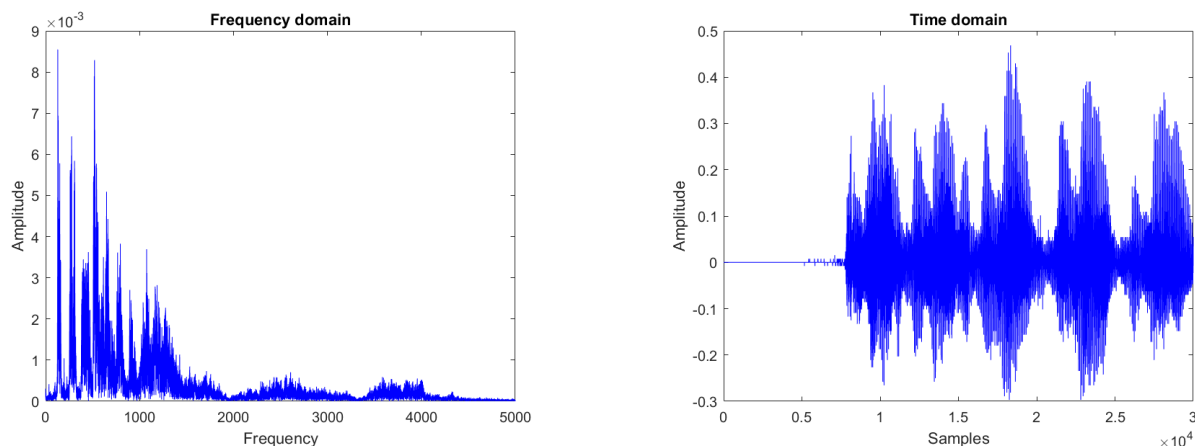
3 Genomförande / Redogörelse för arbetet

För att underlätta processen så delades arbetet upp i tre delar. Del 1 var att få skriva kod som gör så användaren att kunde spela in ett ljudklipp och sedan analysera datan från ljudklippet. Del 2 var att kryptera datan och spara ner det som en ljudfil. Del 3 var att ge användaren möjligheten att dekryptera en krypterad ljudfil.

3.1 Del 1: Spela in ljud i MATLAB

MATLAB har en inbyggd "Audio recorder object" som användes för att spela in ett ljudklipp. Ljud inspelningsobjektet hade ett samplings takt på $fs = 10\text{kHz}$. Inspelningen gjordes med funktionen `recordblocking(readObj, time)`, där `readObj` är inspelningsobjektet och `time` är tidsperioden för inspelningen.

När inspelningen var klar så hämtades datan med funktionen `getaudiodata(readObj)` och lagrade i variabeln `data`. En plot i tidsdomänen och frekvensdomänen gjordes för att analysera datan för ljudklippet som nu var möjligt att manipulera (se Figur 1).



Figur 1: Data för ett ljudklipp

3.2 Del 2: Kryptera ljudklippet

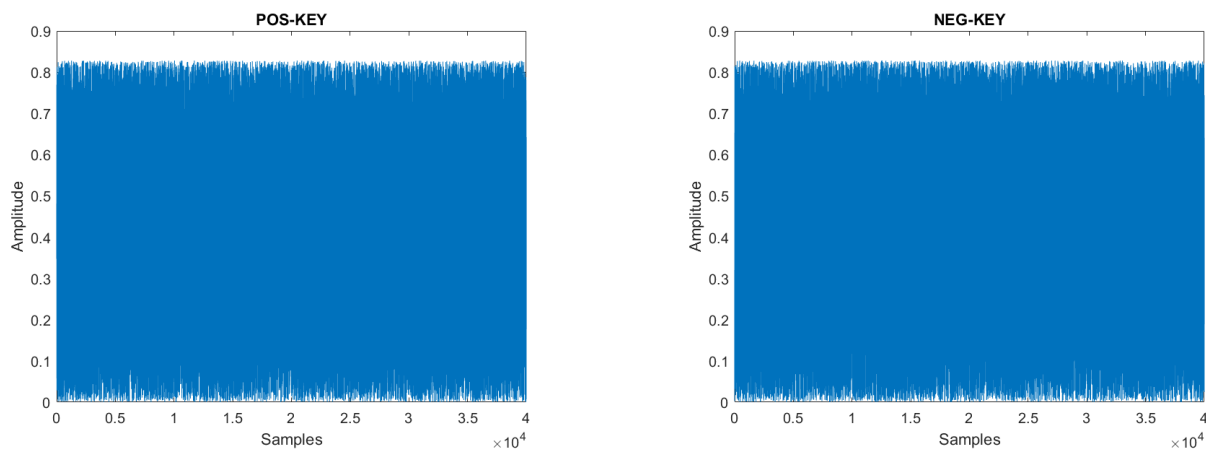
Krypteringen gick till genom att lägga till störningar i `data` med `key = rand(size(data),1)`, där `rand` och `size(data)` är inbyggda MATLAB funktioner som returnerar slumpvisa tal respektive returnerar storleken på `data`. Detta gjorde att `key` blev en vektor med slumpvisa tal och hade samma dimensionsstorlek som `data`. Krypteringen gick helt enkelt till att lägga till och dra ifrån två olika slumpvisa vektorer som kallas för `key_pos` respektive `key_neg` som både gjorde ljudklippet oanvändbart och användes senare till att återställa ljudklippet, därav namnet `key`.

Det uppstod däremot ett problem, analysen av ljudklippet visade att MATLAB sparar endast värden med ett amplitudsintervall mellan $[-1,1]$ så de krypteringsdatan fick inte innehålla värden större eller mindre än 1 respektive -1. Detta löstes genom att användningen av en normerings konstant $k = 1 - \max(data)$, där k är konstanten och $\max(data)$ är en MATLAB funktion som returnerar största värdet i `data`.

Konstanten k multiplicerades med de slumpvisa vektorerna vilket gav följande ekvation:

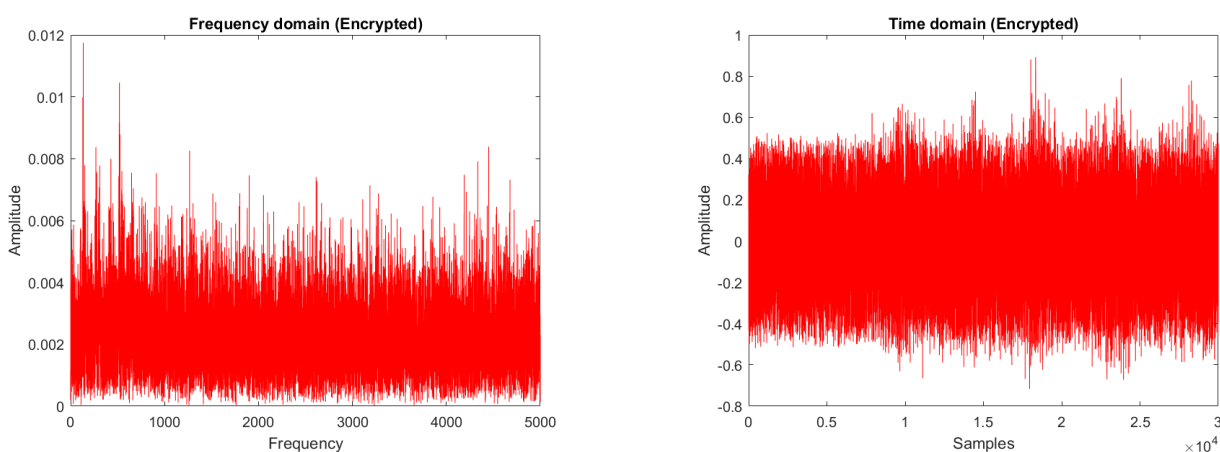
$$data_{krypterad} = data + k \cdot key_{neg} - k \cdot key_{pos} \in [-1, 1]. \quad (1)$$

Anledningen till varför vektorerna kallades för `key_pos` respektive `key_neg` var på grund av att i ekvation 1 så adderas och subtraheras `key_neg` respektive `key_pos` från `data` för att skapa `data_krypterad` (se ekvation 1). Programmet behövde då veta vilken `key` som skulle adderas samt subtraheras under dekrypteringen. Det naturliga var att låta programmet addera `key_pos` för att dekryptera ljudklippet eftersom det subtraherades under krypteringen.



Figur 2: Exempel på key_{pos} och key_{neg}

En plot på den krypterade ljudklippet togs fram för analysering och spelades upp för att säkerställa att ljudklippet är verkligen var oanvändbar (se Figur 3). Den krypterade ljudklippet sparades ner som en .wav fil och slumpstalsvektorererna key_{pos} och key_{neg} sparades som .txt filer som användes för dekrypteringen.



Figur 3: Data för ett krypterad ljudklipp

3.3 Del 3: Dekryptera ljudklippet

Dekrypteringen gjordes genom att använda ekvation 1 och sedan lösa ut $data$ (se ekv. 2).

$$data_{krypterad} = data + k \cdot key_{neg} - k \cdot key_{pos} \Leftrightarrow data = data_{krypterad} - k \cdot key_{neg} + k \cdot key_{pos} \quad (2)$$

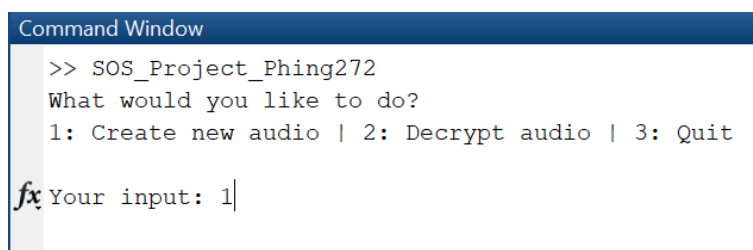
Programmet bad användaren först att skriva in filnamnet på den nersparade krypterade ljudfilen samt *nycklarna*, key_{pos} och key_{neg} textfilerna. Programmet använde ekvation 2 för att lösa ut $data$ och returnerade den dekrypterade ljudklippet. Användaren fick sen höra ljudklippet som därefter sparades ner i en ny .wav fil.

4 Resultat och analys

Varje del sattes ihop och testkördes och fungerade som det skulle. Flera testomgångar kördes för olika ljudklipp såsom låg- eller högljudda ljudklipp för att undersöka om det skulle ge annorlunda resultat vilket kommer att diskuteras i 4.2. Hypotesen om att det inte skulle vara svårt att kryptera en ljudfil stämde som förväntad.

4.1 Resultat: Exempel på testkörning av programmet

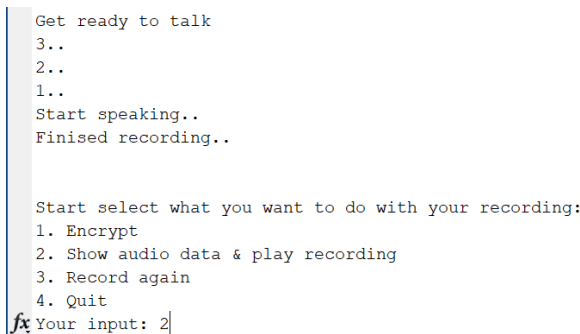
Programmet började med att fråga vad användaren ville göra(se Figur 4). I detta fall valde användaren "1" och skapade ett nytt ljudklipp. Användaren fick bestämma tidsintervallet för inspelningen och spelade in ett ljudklipp.



```
Command Window
>> SOS_Project_Phing272
What would you like to do?
1: Create new audio | 2: Decrypt audio | 3: Quit
fx Your input: 1|
```

Figur 4: *Startval*

Programmet frågade efteråt vad användaren ville göra med ljudklippet, där användaren valde "2" för att lyssna på ljudklippet och visa datan för ljudklippet(se Figur 5).



```
Get ready to talk
3..
2..
1..
Start speaking..
Finised recording..

Start select what you want to do with your recording:
1. Encrypt
2. Show audio data & play recording
3. Record again
4. Quit
fx Your input: 2|
```

Figur 5: *Efter inspelningen*

Därefter ställde programmet samma fråga till användaren igen som i Figur 5. Användaren valde sedan "1" för att kryptera ljudklippet och lyssnade igenom den(se Figur 6).

```

Encrypting Audio...

Play back encrypted audio?
Y = YES | N = NO

Your input: Y

Play back encrypted audio again?
Y = YES | N = NO

fx Your input: N|

```

Figur 6: Efter kryptering

Användaren fick därefter se datan för den krypterade ljudklippet och allt sparades i .wav och .txt filer(se Figur 7).

```

Saving decrypted audio...
Encrypted audio was a success!
Audiofile saved as: EncryptedAudio.wav
Keyfiles saved as: POS_KEY.txt and NEG_KEY.txt

Quitting..
fx >> |

```

Figur 7: Sparad ljudfil och nycklar

Programmet startades om igen för att testa dekryptering. Användaren fick fylla i .wav filen samt de två *nycklarnas* .txt. Ljudfilen dekrypterades med ekvation 2 och sparades ner som en ny .wav fil(se Figur 8).

```

Command Window

>> SOS_Project_Phing272
What would you like to do?
1: Create new audio | 2: Decrypt audio | 3: Quit

Your input: 2
Please enter audio filename: EncryptedAudio.wav
Please enter positive Keyfile: POS_KEY.txt
Please enter negative Keyfile: NEG_KEY.txt
Decrypting audio..
Success!!

What would you like to do?
1. Play decrypted audio & show data
2. Save and quit
fx Your input: |

```

Figur 8: Dekryptering

Fullständiga programkörning och MATLAB kod finns i Bilaga A respektive Bilaga B.

4.2 Analys och slutsatser

Programmet fungerade som den skulle och det var förvånansvärt enkelt att skriva koden eftersom idén bakom krypterings ekvationen (se ekv. 1) var lätt att implementera. Med det sagt så upptäcktes det att programmet hade sina brister.

Till exempel så gick krypteringen ut på att lägga till störningar som skapades av $k \cdot key$, där $k = 1 - \max(data)$ och $key = rand(size(data), 1)$. Om en användare hade spelat in ett ljudklipp med hög amplitud, det vill säga ett amplitud nära 1 så kommer $k = 1 - (\max(data))$ att vara nära 0 vilket ger ett lågt brus. Detta blir inte bra då det inte är stor skillnad mellan den krypterade ljudfilen och den originella. Om programmet ska utnyttjas effektivt bör användaren prata i samtalsnivå eller till och med även viska för bästa resultatet.

Varför det idag inte finns många ljudkrypterings program tror jag är på grund av låg efterfrågan. Vad används egentligen ett krypteringsprogram till för? Jo, för att hålla dölja innehållet av det som skickas. När information skickas så skickas det oftast som en textfil snarare än en ljudfil eftersom det kräver mindre data att skicka en textfil än en ljudfil. Det är möjligt att detta är anledningen till det låga efterfrågan på just ljudkrypterings-program.

Slutsatsen är att ett ljud krypteringsprogram är inte nödvändigtvis svårt att skapa i MATLAB. Någon med grundläggande kunskaper inom MATLAB skulle klara av det men med stor sannolikhet så skulle det förmodligen inte komma till användning på grund av den låga efterfrågan.

Referenser

- [1] The MathWorks, Inc
<http://https://se.mathworks.com/>

A Körning av programmet

Command Window

```
>> SOS_Project_Phing272
What would you like to do?
1: Create new audio | 2: Decrypt audio | 3: Quit

Your input: 1
Time duration for the recording? (Minimum = 1 sec | Maximum = 10 sec)
Your input: 3
Get ready to talk
3..
2..
1..
Start speaking..
Finised recording..

Start select what you want to do with your recording:
1. Encrypt
2. Show audio data & play recording
3. Record again
4. Quit
Your input: 2

Generating data..

Start select what you want to do with your recording:
1. Encrypt
2. Show audio data & play recording
3. Record again
4. Quit
fx Your input: 1
```

Encrypting Audio...

Play back encrypted audio?

Y = YES | N = NO

Your input: Y

Play back encrypted audio again?

Y = YES | N = NO

Your input: N

Show encrypted audio data?

Y = YES | N = NO

Your input: Y

Saving decrypted audio...

Encrypted audio was a success!

Audiofile saved as: EncryptedAudio.wav

Keyfiles saved as: POS_KEY.txt and NEG_KEY.txt

Quitting..

fx >> |

Command Window

```
>> SOS_Project_Phing272
```

```
What would you like to do?
```

```
1: Create new audio | 2: Decrypt audio | 3: Quit
```

```
Your input: 2
```

```
Please enter audio filename: EncryptedAudio.wav
```

```
Please enter positive Keyfile: POS_KEY.txt
```

```
Please enter negative Keyfile: NEG_KEY.txt
```

```
Decrypting audio..
```

```
Success!!
```

```
What would you like to do?
```

```
1. Play decrypted audio & show data
```

```
2. Save and quit
```

```
fx Your input: |
```

B Fullständig MATLAB kod

```
clear all

readObj = audiorecorder(10000,8,1); % Create audio recorder
newAudioChoice = 0;
fs = 10000; %Sampling rate

%Ask user what they want to do
disp(['What would you like to do?' newline '1: Create new audio | 2: Decrypt audio | 3: Quit' newline ])
userInput = 'Your input: ';
answer = input(userInput);

%===== 1 Create new audio =====
while 1 && answer == 1;

disp('Time duration for the recording? (Minimum = 1 sec | Maximum = 10 sec)');
timeInput = 'Your input: ';
time = input(timeInput);

%Time boundary check
if time >= 10
    time = 10;
end
if(time <= 1)
    time = 1;
end

%The countdown
disp('Get ready to talk'); pause(3);
disp('3..'); pause(1); disp('2..'); pause(1); disp('1..'); pause(1);

%Start recording
disp('Start speaking..');
recordblocking(readObj, time);
disp('Finised recording..');
disp(newline);
disp('Start select what you want to do with your recording: ');
disp(['1. Encrypt' newline '2. Show audio data & play recording' newline '3. Record again' newline '4. Quit']);
data = getaudiodata(readObj);

%Get user input for what they want to do
prompt = 'Your input: ';
newAudioChoice = input(prompt);

while(newAudioChoice == 2)
    %Play audio
    soundsc(data,fs);

    pause(2);
    disp([newline 'Generating data..' newline]);

    %Plot the time domain
    figure(1)
```



```
plot(data, 'blue');
title('Time domain');
xlabel('Samples');
ylabel('Amplitude');

%Plot the frequency domain
figure(2)
Y = fft(data);
N = length(Y);
f = 0:fs/N:fs/2;
X_magn = abs(Y)/(N/2);
plot(f, X_magn(1:length(f)), 'blue');
title('Frequency domain')
xlabel('Frequency');
ylabel('Amplitude');

%Ask again
disp('Start select what you want to do with your recording: ');
disp(['1. Encrypt' newline '2. Show audio data & play recording' newline '3.✔
Record again' newline '4. Quit']);

prompt = 'Your input: ';
newAudioChoice = input(prompt);
end

%If x == 3 record again
if(newAudioChoice ~= 3)
    break;
end

end

%1. Encrypt audio
if(newAudioChoice == 1)

    disp('Encrypting Audio...');
    pause(2);

    %Find value closest to 1
    distance = max(data);

    POS_KEY = (1-distance)*rand(time*fs,1); %Generate Positive KEY
    NEG_KEY = (1-distance)*rand(time*fs,1); %Generate Negative KEY

    data_encrypt = data - POS_KEY + NEG_KEY;

    %Ask user for audio playback
    disp([newline 'Play back encrypted audio?' newline 'Y = YES | N = NO' newline]);
    userInput2 = 'Your input: ';
    playBackChoice = input(userInput2, 's');

while playBackChoice == 'y' || playBackChoice == 'Y';

    %Play audio
```

```
    soundsc(data_encrypt, fs);

    %Ask user for audio playback again
    disp([newline 'Play back encrypted audio again?' newline 'Y = YES | N = NO'↵
newline]);
    userInput2 = 'Your input: ';
    playBackChoice = input(userInput2, 's');

end

%Ask user if they want to see encrypted audio data
disp([newline 'Show encrypted audio data?' newline 'Y = YES | N = NO' newline])
userInput3 = 'Your input: ';
showEncryptedDataChoice = input(userInput3, 's');

if(showEncryptedDataChoice == 'y' || showEncryptedDataChoice == 'Y')

    %Plot encrypted audio in time domain
    figure(3)
    plot(data_encrypt, 'red');
    title('Time domain (Encrypted)')
    xlabel('Samples');
    ylabel('Amplitude');

    %Plot encrypted audio in frequency domain
    figure(4)
    Y = fft(data_encrypt);
    N = length(Y);
    f = 0:fs/N:fs/2;
    X_magn = abs(Y)/(N/2);
    plot(f, X_magn(1:length(f)), 'red');
    title('Frequency domain (Encrypted)')
    xlabel('Frequency');
    ylabel('Amplitude');
end

filename_audio = 'EncryptedAudio.wav';
filename_POS_KEY = 'POS_KEY.txt';
filename_NEG_KEY = 'NEG_KEY.txt';

%Write encrypted audio and key to file
audiowrite(filename_audio, data_encrypt, fs);
writematrix(POS_KEY, filename_POS_KEY);
writematrix(NEG_KEY, filename_NEG_KEY);

disp(newline);
disp('Saving decrypted audio...');
pause(2);
disp('Encrypted audio was a success!');
disp('Audiofile saved as: EncryptedAudio.wav');
disp('Keyfiles saved as: POS_KEY.txt and NEG_KEY.txt');
disp(newline);
```

```
disp('Quitting..');

end

%4. Quit
if(newAudioChoice == 4 || answer == 3)
    disp('Quitting..');
end

%===== 2. Decrypt Audio =====
if answer == 2

    %Get encrypted audio and text files
    fileinput = 'Please enter audio filename: ';
    filename = input(fileinput, 's');

    posKeyRequest = 'Please enter positive Keyfile: ';
    posKeyInput = input(posKeyRequest, 's');

    negKeyRequest = 'Please enter negative Keyfile: ';
    negKeyInput = input(negKeyRequest, 's');

    [y, fs] = audioread(filename);

    fileID_POS = fopen(posKeyInput, 'r');
    formatSpec_POS = '%f';
    posKey = fscanf(fileID_POS, formatSpec_POS);
    fclose(fileID_POS);

    fileID_NEG = fopen(negKeyInput, 'r');
    formatSpec_NEG = '%f';
    negKey = fscanf(fileID_NEG, formatSpec_NEG);
    fclose(fileID_NEG);

    %Decrypt audio using keys
    DecryptedAudio = y + posKey - negKey;

    disp('Decrypting audio..');
    pause(2);
    disp('Success!!');
    pause(1);

    disp(newline);
    disp(['What would you like to do?' newline '1. Play decrypted audio & show data'↵
newline '2. Save and quit']);

    DecryptQuestion = 'Your input: ';
    DecryptChoice = input(DecryptQuestion);

    %Play audio and show data.
    if DecryptChoice == 1

        soundsc(DecryptedAudio, fs);
```

```
disp('Generating data..');

%Plot the time domain
figure(1)
plot(DecryptedAudio, 'blue');
title('Time domain');
xlabel('Samples');
ylabel('Amplitude');

%Plot the frequency domain
figure(2)
Y = fft(DecryptedAudio);
N = length(Y);
f = 0:fs/N:fs/2;
X_magn = abs(Y)/(N/2);
plot(f, X_magn(1:length(f)), 'blue');
title('Frequency domain')
xlabel('Frequency');
ylabel('Amplitude');

secondAnswer = 'Y';

%Play audio again?
while secondAnswer == 'Y' || secondAnswer == 'y';

secondQuestion = 'Do you want to hear that again? Y = Yes | N = No : ';
secondAnswer = input(secondQuestion, 's');

if(secondAnswer == 'y' || secondAnswer == 'Y')
    soundsc(DecryptedAudio, fs);
end

end

end

%Save to file
audiowrite('Decrypted_Audio.wav', DecryptedAudio, fs);

disp('Saving and exiting..')
pause(2);

end
```