EE323 Digital Signal Processing
Assignment - II Report

Team Members: Swayam Borate(23110066),Parth Dembla(23110234), Shriniket Behera(23110306)

In the previous assignment, a secure and intelligent speech communication Android application was developed in Simulink using **Method 1: Frequency Scrambling with Key-Based Permutation**. In this approach, the frequency components of the recorded speech signal were scrambled using a key-based mapping generated through a pseudo-random number generator that implements the Fisher-Yates algorithm, making the audio unintelligible without the correct key. The key was configurable within the application and was required at the receiver end for successful decryption.

In the current assignment, two additional encryption/authentication methods are integrated into the same mobile application to enhance flexibility and security. **Method 2 (Noise Masking with Adaptive Noise Shaping)** introduces synthetic noise, combined with adaptive spectral shaping, to obscure the speech signal in real-time. **Method 3 (Speech Watermarking for Authentication)** embeds an inaudible watermark into the audio for receiver-side verification before decryption. Both new methods are selectable from the application's UI, allowing the user to choose the desired speech protection mechanism.

Objectives:-

1. Implement **Method 2: Noise Masking with Adaptive Noise Shaping**, where synthetic noise (white noise, tones, or shaped noise) is added to mask the speech, along with an adaptive noise-shaping filter that adjusts noise characteristics based on the speech spectrum.

2. Implement **Method 3: Speech Watermarking for Authentication**, which involves embedding a low-level, inaudible watermark into the speech signal and enabling watermark extraction at the receiver to verify authenticity before decryption.

3. Display authentication results (pass/fail) in the UI.

4. Develop a corresponding **Phone B application** capable of receiving encrypted/authenticated audio over Bluetooth, selecting the appropriate decryption method, and reconstructing the original speech from any of the three methods.

5. Integrate a **signal quality assessment module** (SNR or PESQ) to estimate the quality of the decrypted audio and display this metric on the UI after decryption.

Method 2:- <u>Noise Masking with Adaptive Noise Shaping</u>

The idea is to add noise that is spectrally shaped to resemble speech, so the signal becomes unintelligible but still sounds speech-like. In decryption, noise is removed using the same estimated shape.

The Structure of the Sender and receiver modules remains same as done previously

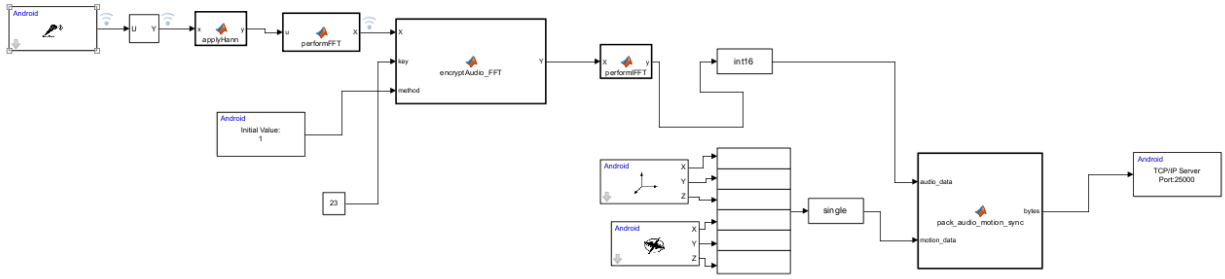


Figure-1: Sender Model

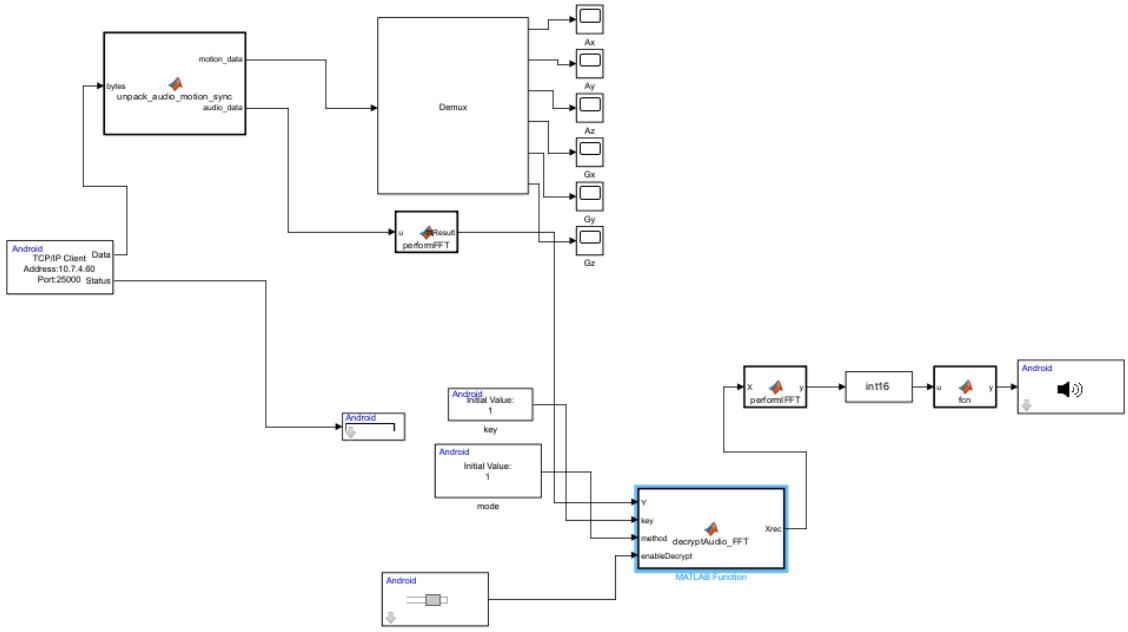


Figure-2: Receiver Model

The Change is in encryption(in the sender's module) and decryption(in the receiver's module) blocks MATLAB functions

The steps followed for this method were:-

1. The FFT of the speech signal in the time domain is taken to convert it to the frequency domain, and its magnitude and phase responses were obtained.

2. The speech magnitude spectrum is smoothed using averaging. This removes rapid fluctuations and retains the overall shape. Noise shaping needs a smooth estimate of speech energy per frequency band.

Why smoothening?

The **spectral envelope** is a smooth curve that represents the overall distribution of energy across different frequencies in a speech signal. Instead of examining the fine, rapidly changing FFT magnitudes caused by harmonics, the envelope captures the broad shape of the spectrum, indicating where speech has strong energy and where it has weak energy. It is essentially a smoothed version of the magnitude spectrum that reflects the general frequency structure of speech. We need the spectral envelope in noise shaping because adding raw white noise does not effectively mask speech; it has equal energy at all frequencies, and sounds unnatural. By shaping the noise according to the speech's spectral envelope, the noise gains a similar energy pattern to the speech, making it blend naturally and mask the important speech components. This ensures the encrypted audio becomes unintelligible while still sounding speech-like, providing more effective and perceptually balanced masking. It is done using `magEnv = movmean(magX, 8, 1);` it takes a **moving average** over 8 neighbouring frequency bins.

3. White noise originally has equal energy across all frequencies, but speech does not. Therefore, the noise must be "shaped" using the speech's spectral envelope to match its energy pattern, which makes the noise blend with speech and mask it effectively. This is done using `noise_shaped = noise .* (magEnv ./ max(magEnv(:) + eps))` where noise is the generated white noise using randn() function

4. Adaptive noise shaping multiplies the white noise by the speech's spectral envelope, boosting noise in frequency regions where speech has high energy and reducing it where speech is weak, so the noise copies the speech's spectral shape and effectively hides the real content.

```
% --- 4) Adjust masking level ---
noise_level = 0.2 * mean(magX(:));   % 0.1–0.3 typical range
noise_shaped = noise_shaped * noise_level;
```

5. The noise level is set as a fraction of the average speech energy so that the added shaped noise is strong enough to make speech unintelligible but not so strong that it causes harsh distortion, typically using 10–30% of the speech magnitude.

6. The shaped noise is added only to the speech magnitude, while the phase remains unchanged. As a result, the receiver re-estimates the spectral envelope from the encrypted signal using `movmean()`, which smooths the magnitude of the received FFT (`abs(Y)`) to recreate the overall energy shape of the original speech. The final encrypted FFT has its magnitude masked by noise but retains the original phase, making the audio unintelligible yet still reconstructable.

7. For Decryption, the receiver re-estimates the spectral envelope from the encrypted signal **using** `movmean()`, which smooths the magnitude of the received FFT (`abs(Y)`) to recreate the overall energy shape of the original speech.

8. New white noise is generated with `randn()` and shaped using the same spectral envelope, by multiplying the noise with the normalized envelope (`noise * (magEnv/max(magEnv))`) so that its frequency structure resembles the noise added during encryption.

9. The shaped noise is scaled to the same approximate masking level using a multiplier (`noise_level = 0.8 * mean(magY(:))`), recreating a noise pattern similar to what was added at the transmitter.

10. The receiver subtracts this shaped noise from the encrypted magnitude (`magRec = magY - noise_shaped`) and reconstructs the FFT using the original phase (`exp(1j*phaseY)`), producing an approximate recovery of the original speech.

Summary:-

The noise-masking method works by converting the speech signal to the frequency domain, smoothing its magnitude spectrum using a moving average to extract the spectral envelope, which represents the overall distribution of energy across frequencies. This envelope is important because raw white noise has equal energy at all frequencies and does not mask speech effectively. By generating white noise and shaping it according to this spectral envelope, the noise acquires a frequency pattern similar to that of the speech signal, allowing it to blend naturally and hide intelligible components while still sounding speech-like. The shaped noise is then scaled and added only to the magnitude of the speech while preserving the phase, producing an unintelligible but reconstructable encrypted signal. During decryption, the receiver re-estimates the spectral envelope, regenerates similarly shaped noise, and subtracts it from the encrypted magnitude before reconstructing the FFT, yielding an approximate recovery of the original speech. This technique is useful in secure voice communication, speech anonymization, and low-complexity privacy systems, as the noise is adapted to the speech structure, making unauthorized recovery far more difficult.

Method 3:- <u>Speech Watermarking for Authentication</u>

In the watermarking method, a low-energy, inaudible identification signal (called a watermark) is embedded into specific frequency bins of the speech spectrum, allowing the receiver to verify that the audio truly originated from an authorized sender. The watermark is created by generating a pseudo-random noise (PN) sequence using a known secret key and adding this sequence, scaled by a very small factor to the magnitude of the speech FFT, only within a safe mid-frequency band (1-4 kHz), where it remains inaudible but detectable. At the receiver, the same PN sequence is regenerated using the same key, and the extracted spectral region is correlated with this known pattern. If the correlation is strong, it means the watermark is present and has not been altered, so the signal is authentic, and authentication is set to 1. If the correlation is weak or missing, it indicates tampering or an untrusted source, so authentication is considered invalid, and decryption is not performed. In simple terms, watermarking is the process of hiding a unique, secret signature within the audio, allowing the receiver to verify its legitimacy before trusting or decrypting the content.

For this an embed watermark function block was created on seder side and extract watermark was created on receiver's side

On Sender's Side:-

- A low-level, inaudible watermark is added to the speech spectrum by modifying only the magnitude of the FFT in a selected frequency band (1–4 kHz), obtained using find(f >= bandStart & f <= bandEnd), so that the watermark does not affect audible quality.

- A pseudo-random noise (PN) sequence is generated using rng(key) and randn(), where the key controls the seed and ensures the watermark pattern is secret and repeatable at the receiver; this PN sequence acts as the hidden signature.

- The watermark is embedded by adding a scaled version of the PN sequence to the magnitude (magX(bandIdx) += alpha * pn), where alpha controls the watermark strength, ensuring it remains inaudible but detectable.

- The phase is kept unchanged, and the FFT is reconstructed using magX* exp(1j * phaseX), so the audio sounds identical but contains a hidden authentication pattern.

On Receiver's Side:-

- The receiver regenerates the same PN sequence using the same key with rng(key) and randn(), ensuring it produces the identical watermark pattern used by the sender.

- The magnitude from the same band (1–4 kHz) is extracted using magY = mean(abs(Y),2) and bandIdx = find(), isolating the region where the watermark was embedded.

- Authentication is performed by computing the correlation (corrVal = sum(magY_band .* pn)), where a high correlation means the watermark is present → audio is authentic, while a low

correlation means missing/tampered signal → authentication fails.

- Authentication output becomes 1 only when the correlation exceeds a threshold, meaning the watermark matches the expected PN pattern; otherwise authentication is 0, preventing decryption of untrusted audio.

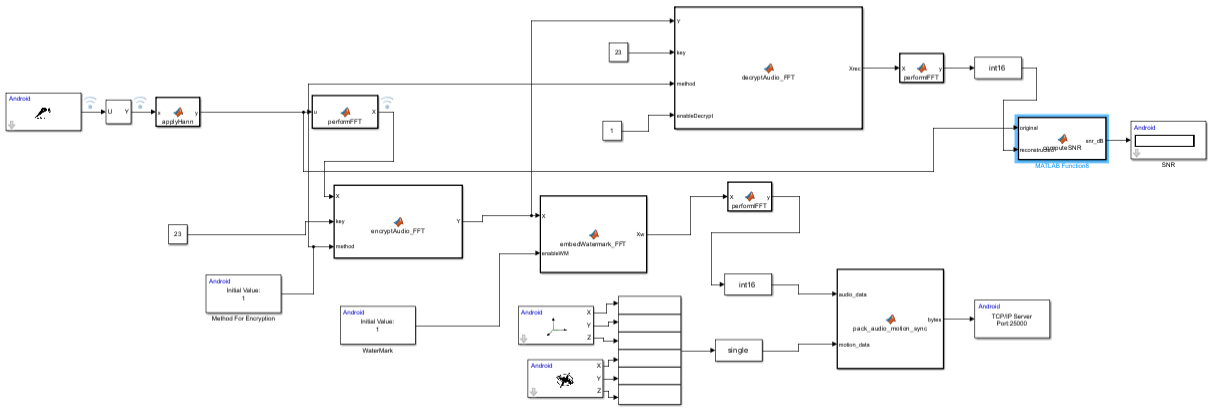
Summary: The watermarking method embeds a low-level, inaudible authentication signal into the speech by modifying the FFT magnitude only within a controlled frequency band. A pseudo-random noise (PN) sequence is generated using a secret key (rng + randn), normalized, and added in a scaled form to the speech magnitude so that the watermark remains hidden but detectable. At the receiver, the same PN sequence is regenerated using the same key, and the corresponding frequency band is extracted from the received signal. The receiver then computes the correlation between the extracted magnitudes and the expected PN pattern; a high correlation indicates that the watermark is present and the audio is authentic, so authentication is set to 1. If the correlation is low or mismatched, authentication becomes 0, meaning the audio has been altered or is not from a trusted sender. In essence, watermarking provides a secure way to verify the legitimacy of the speech signal before allowing decryption.

Next, we needed to show the SNR of the transmitted signal

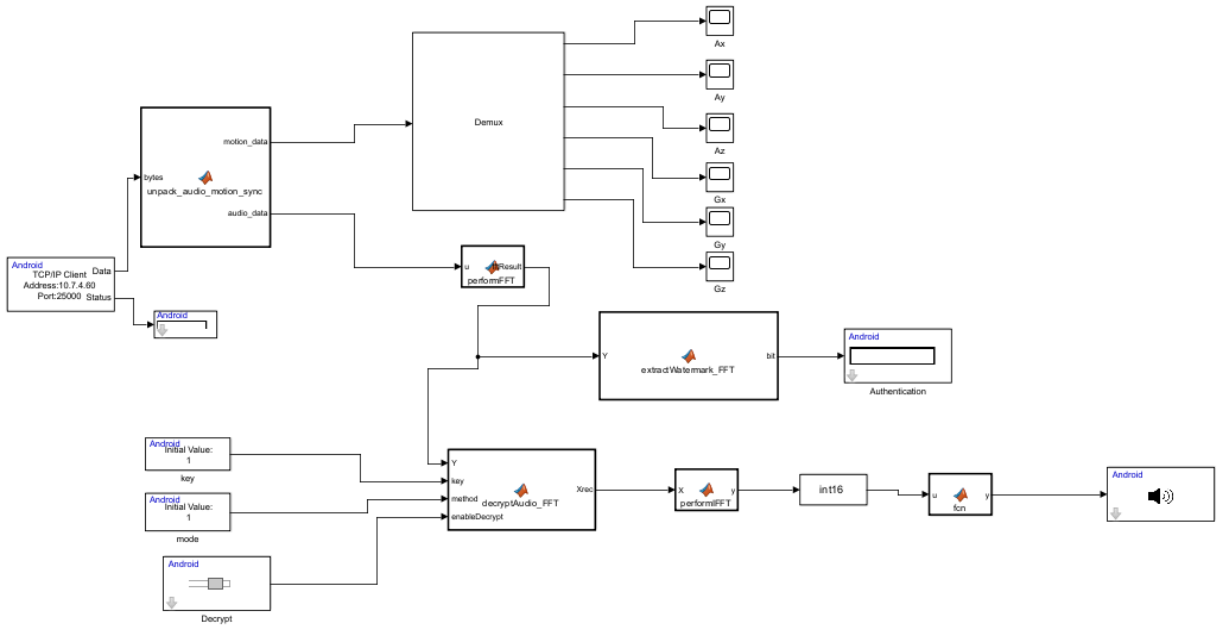
To implement this, the signal processing that was done on the receiver's side is also performed on the sender's side. With the SNR block, the formula is integrated, and the SNR is computed.

figures below show the full Senders and Reciever’s Modules

Sender:



Receiver:



We observed that the SNR for Method 1 (key-based frequency permutation) was very high, typically around 60–70 dB. This is expected because this method does not introduce any additional noise into the signal; it simply permutes the frequency bins in a reversible manner. Since the original magnitudes are preserved and the permutation can be perfectly inverted during decryption, the reconstructed signal is almost identical to the original, resulting in very low error power and, consequently, a high SNR. In contrast, the SNR for Method 2 (noise masking with adaptive noise shaping) was significantly lower, around 45 dB. This is because Method 2 intentionally adds shaped noise to the speech magnitude spectrum in order to mask intelligible content. During decryption, a new shaped noise signal is regenerated and subtracted, but it cannot perfectly match the exact noise added during encryption, making the process inherently non-reversible. As a result, residual noise remains in the reconstructed signal, increasing the error power and reducing the overall SNR.

A demonstration video is also made, demonstrating the working of the app on both devices

Click This Youtube Link! Or Scan QR code