Biometric Authentication using ECG Signal

R S Jaidev (CB.EN.U4CCE22036)
Satyajit A V (CB.EN.U4CCE22040)
Amrish Seenu R (CB.EN.U4CCE22061)
Department of Electronics and Communication Engineering,
Amrita School of Engineering, Coimbatore,
Amrita Vishwa Vidyapeetham, India

Abstract:

In this paper, a Biometric Authentication system has been simulated using electrocardiogram (ECG) signals as bio-metrics. In this work, we propose a two-phase method to conduct human authentication using the ECG signal, which are the feature extraction and the classification. In the first phase, we have used Band-pass filter and feature extraction tech techniques such as Symmlet and Daubechies.In the second phase, Dynamic Time Warping (DTW) was proposed to compare two sequence and have computed the Evaluation Metrics which comprehensive metrics such as accuracy, false accept rate (FAR), false reject rate (FRR), and equal error rate (EER) have been evaluated with the databases.The proposed methods are evaluated using a public databases namely ECG-ID database obtained from the Physionet database. Experimental results show that our features can achieve high subject identification accuracy of 100% on ECG signals that are from ECG-ID (Five recording),indicating that our features makes it possible to improve the efficiency of our authentication system.

Keywords-Biometric Authentication; ECG signal ; Band-pass Filter ; Symmlet and Daubechies ; DTW ; FAR ; FRR ; EER .

I. EXPLORING ELECTROCARDIOGRAM (ECG) SIGNALS AS ROBUST BIOMETRIC IDENTIFIERS

Bio-metrics, the identification of individuals using their physical or behavioral traits, has become a common topic in our daily conversations. Various biometric methods exist, including fingerprint, Speech recognition, facial recognition, iris scanning, and more. However, as technologies for falsification advance, these characteristics can be counterfeited. Hence, there's a growing need to explore new features that are harder to replicate, such as physiological signals like the Electrocardiogram (ECG).

There's a pressing need to create a human authentication system that can eradicate issues stemming from fraudulent methods. When selecting a biometric for human identification or authentication, certain fundamental criteria must be met: it should be universally possessed, unique to each individual, and permanently accessible to the person. Electrocardiogram (ECG), also known as a record of the heart's electrical signals, fulfills these essential criteria for a biometric consideration in human authentication. ECG stands out as a potentially efficient biometric due to numerous advantages, offering insights into an individual's health status each time it's employed for verification.

II. RELATED WORK

[1]Sihem Hamza and Yassine Ben Ayed worked on usion of three new types of characteristics: cepstral coefficients, ZCR, and entropy. In the second phase, the support vector machines (SVM) has been applied for the classification system. The proposed methods are evaluated using two public databases namely MIT-BIH arrhythmia and ECG-ID database obtained from the Physionet database.

In [3], an ECG-based authentication system suitable for security protocols and medical facilities is described. The performance of the proposed system was evaluated and the identification accuracy was found to be 92%. The authentication approach is implemented using machine learning algorithms that provide better performance when working with ECG data. The suggested system explores datasets and obtained high-quality data for ECG-based biometric authentication.

Article [5] explores a new biometric based on ECG signal. For exploring the reliability of ECG as a biometric, in this research, they collected data from 55 individuals two times in 4 months. Here they also used consumer-grade ECG monitors that were cheaper and easier to use than their medical counterparts. Based on the experimental results, it has been demonstrated that the ECG signals taken using readily available consumer-grade monitors can effectively be utilized in the identification process. The steps in this methodology include dataset collection, preprocessing dataset, classification, and performance evaluation. ECG signals are preprocessed by removing baseline wander and high-frequency noise. A bandpass filter is used to isolate the QRS complex of the ECG signal. The extracted QRS complex is used for extraction features such as mean, standard deviation, variance, and energy. A template is created with features extracted by the user. The template represents the user's ECG signal. During the authentication process, the user's ECG signal is compared to a template. Here the working of the system is evaluated using various metrics such as false reception rate (FAR), false reject rate (FRR), and receiver performance curve (ROC). The proposed system shows promise in terms of accuracy and performance, making it a potential candidate for biometric authentication. The results presented in this paper offer a positive outlook for ECG-based biometrics and show that individuals can be authenticated through ECG recordings. The study has found that performance degradation of the ECG biometric system over time is consistent with previous works. Although, it also shows that improvements can be made to the

performance by adapting the stored biometric data with new signals after successfully authenticating the individual[4].

III. METHODOLOGY

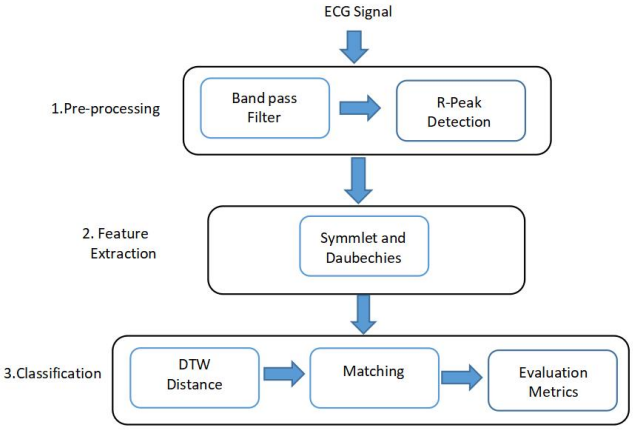


Fig. 1. Architecture of the ECG identification methodology proposed.

1.1 Pre-processing

The ECG signal is acquired from the electrodes placed on the limbs.The Pre-processing phase is necessary to remove the noises. Therefore, the Pre-processing operation has three stages, namely filtering, R-peak value. We applied the band-pass filter (2-50 Hz) for each ECG record to eliminate the baseline wander and the power line interference. After this operation, we applied the normalization

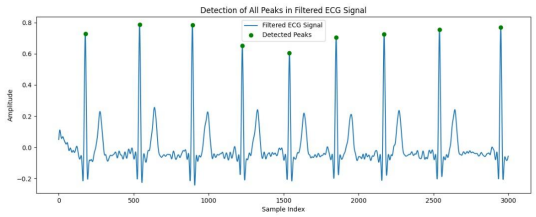
1.1.1 Band pass Filter

ECG signal undergoes the preprocessing phase. Therefore, the preprocessing operation we have used is Bandpass filtering (2-50 Hz), which involves isolating the frequencies of interest that correspond to the heart's electrical activity while eliminating unwanted frequencies (such as noise or interference) that might be present in the signal. This process helps in better analyzing and interpreting the ECG signal.(Using butter-worth filter order of 4^{th} order

$$H(z) = \frac{K \cdot (z - z_1)(z - z_2)\ldots(z - z_{2N})}{(z - p_1)(z - p_2)\ldots(z - p_N)}$$

1.1.2 R-peak Detection

After having corrected the baseline and filtering the ECG signal, the R peaks detection is effect by adaptive thresholding. The objective of R peak detection is to locate the timing position for all true R positive peaks while removing false positive peaks.



.Fig. 2.R-Peak of ECG Signal

1.2 Feature Extraction

The feature extraction stage translates the segmented ECG into a representation that further reduces the effects of intra-subject variability while emphasizing discriminative and intra-class variations to obtain better performance using non-fiducial point.

1.2.1 Non-fiducial point

In the non-fiducial feature extraction technique, the features of focus are the holistic analysis of an ECG, typically con sisting of applying time or frequency analysis to obtain other statistical features. In this paper, we used different wavelet families, such as Symmlet and Daubechies. The reason for using Symmlet and Daubechies mother wavelet is that the function is similar to the ECG signal. In order to evaluate the effectiveness of the aforementioned mother wavelet transformations, different levels of decomposition have been examined, however, we only report a level of four decompositions.

1.3 Classification

1.3.1 Matching

In the matching stage, identification and verification functions can be performed. The purpose of matching is to compare the query ECG feature sets against stored templates to generate match scores. The matching score is a quantitative measurement that checks the similarity between template and query ECG feature sets. A Higher match score indicates that the template and query have a high correlation. In this paper, we will be using Dynamic Time Warping (DTW).

1.3.1.1 Dynamic Time Warping (DTW)

The Dynamic Time Warping (DTW) technique is utilized for comparing sequences of varying lengths, enabling optimal alignment between them by minimizing the sum of differences between corresponding points. This method is particularly effective for comparing unaligned sequences like fingerprints or those with differing lengths, such as in ECG analysis. In our study, we utilize RR segmentation based on heart rate variability, resulting in distinct segments. Consequently, DTW plays a crucial role in measuring similarity between ECG feature sets of template and query after aligning them, facilitating accurate comparison.

1.3.2 Evaluation metrics

1.Error Rate (ERR):Overall rate of incorrect classifications or detection made by an algorithm.

2.False Acceptance Rate (FAR):Rate of incorrectly accepting an unauthorized input as a valid match.

3.False Rejection Rate (FRR):Rate of incorrectly rejecting a valid input or failing to recognize a match.

4.Receiver Operating Characteristic (ROC) curve: Graphical tool to evaluate a system's ability to discriminate

between different classes by varying decision thresholds, plotting Sensitivity vs. 1-Specificity.

These metrics and ROC analysis help assess the accuracy and effectiveness of ECG signal processing algorithms for tasks like anomaly detection or classification.

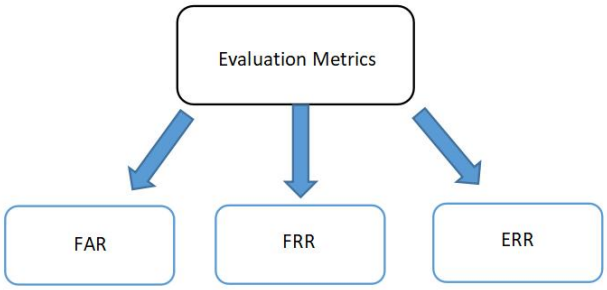


Fig. 3. Evaluation Metrics

IV. Results and Discussions

4.1 Data-set Description

ECG identification database (ECG-ID) were recorded for biometric identification purpose [59]. Each raw ECG record was acquired for about 20 seconds with a sampling rate of 500 Hz and a 12-bit resolution. The first two records acquired from the same day were used for each subject. The database consists of 310 one-lead ECG recording sessions obtained from 90 volunteers during a resting state. The number of sessions for each volunteer varied from 2 to 20 with a time span of 1-day to 6-months between the initial and last recordings. The challenges in this database are the number of noisy environment condition in which two records such as filtered and noisy ECG signal mimics real-world scenarios.

4.2 Performance metrics

Dataset	Filtering	Feature Extraction				
		Non-Fiducial				
		FAR	FRR	ERR	Accuracy	
		DTW	DTW	DTW	DTW	
ECG-ID (90 Users)	Band-pass Filter	1.0	0.1999	0.8	100%	Imposter
		0.2	0	0.6	100%	Genuine user

Table 1. Performance metrics

4.3 Simulation results in figures and tables

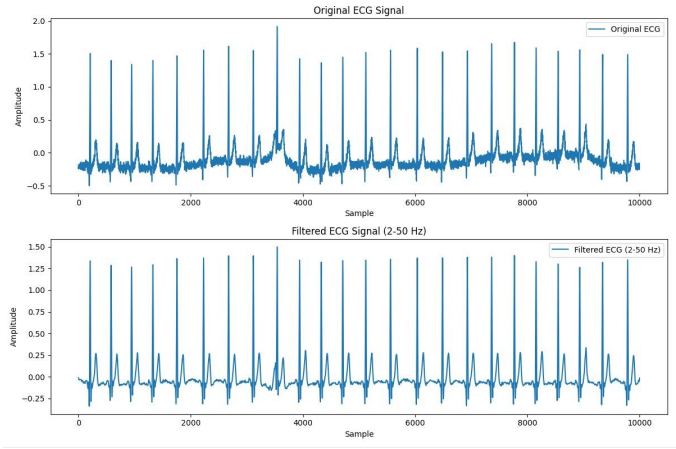


Fig. 4. Filtering Of ECG Signal

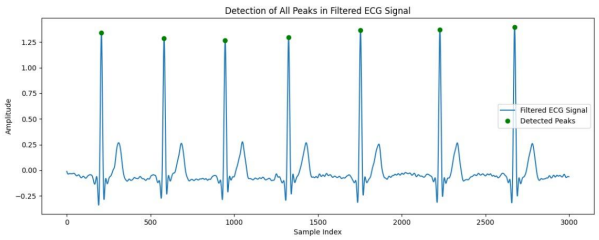


Fig. 5.R-peak detection of ECG Signal.

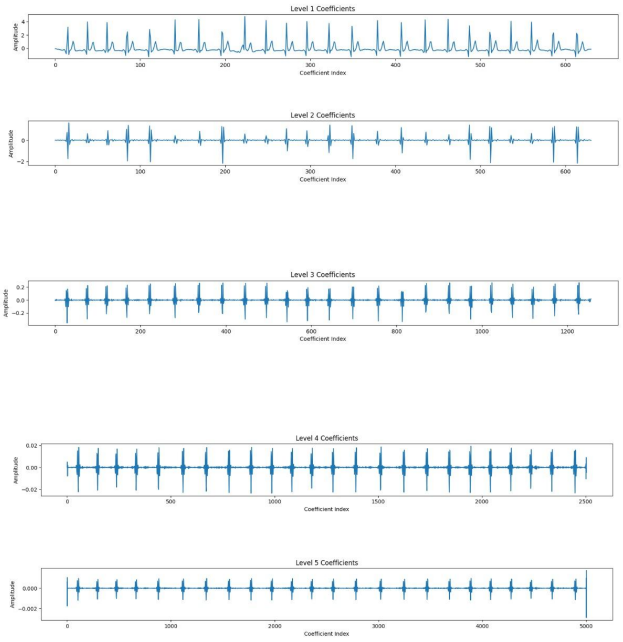


Fig. 6.Different Level of Wavelet decomposition.

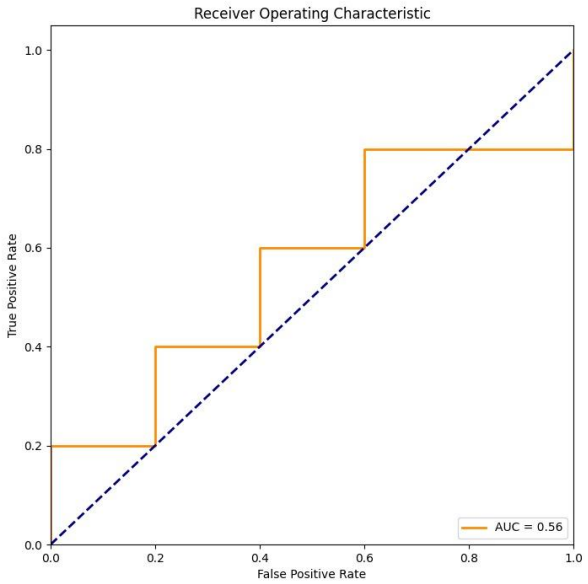


Fig. 7.Receiver operating characteristic

4.4 Inferences

The provided ECG signal processing pipeline aims to enhance the analysis and interpretation of ECG signals. It employs band-pass filtering, R-peak detection, wavelet-based feature extraction, and DTW-based matching. The evaluation metrics, including ERR, FAR, FRR, and ROC analysis, are utilized to assess the accuracy and effectiveness of the algorithm in tasks such as anomaly detection or classification. Overall, the approach is designed to improve the reliability of ECG signal processing for medical applications.

Conclusion

In conclusion, the ECG signal processing pipeline described in the provided text demonstrates a systematic approach to enhancing the accuracy and reliability of ECG signal analysis. The inclusion of pre-processing techniques, such as band-pass filtering and R-peak detection, addresses common challenges like baseline wander and interference. Feature extraction using non-fiducial points, specifically wavelet families like Symmlet and Daubechies, contributes to a holistic analysis of ECG signals.

The choice of Dynamic Time Warping (DTW) for matching ensures effective comparison, particularly for sequences of varying lengths or unaligned sequences. The evaluation metrics, encompassing error rate, false acceptance rate, false rejection rate, and the ROC curve, provide a comprehensive assessment of the algorithm's performance.

This pipeline is well-suited for applications requiring accurate ECG signal processing, such as anomaly detection or classification. The multi-stage approach, involving both preprocessing and advanced feature extraction techniques, positions the algorithm as a valuable tool for medical applications where the precision of ECG analysis is critical for diagnosis and patient monitoring.

Acknowledgment

I express my gratitude to The authors Mohit Ingale and his team at San Jose State University, San Jose; Sihem Hamza and Yassine Ben Ayed at the University of Sfax, Tunisia; and PhysioNet Support at MIT for providing the ECG-ID 1.0.0 dataset.

References

[1] Saroj Kumar Pandey, Rekh Ram Janghel, Vyom Vani, Patient Specific Machine Learning Models for ECG Signal Classification,Procedia Computer Science,2020.

[2] M. Ingale, R. Cordeiro, S. Thentu, Y. Park and N. Karimian, "ECG Biometric Authentication: A Comparative Analysis," in IEEE Access, vol. 8, pp. 117853-117866, 2020, doi: 10.1109/ACCESS.2020.3004464.

[3] P. Poludasu, S. J. Bandaru, A. Kumar Paleru, S. Venkatadas and P. A K, "Biometric Authentication using ECG Signals," 2023 4th International Conference for Emerging Technology (INCET), Belgaum, India, 2023, pp. 1-6, doi: 10.1109/INCET57972.2023.10170104.

[4] AL Goldberger, LAN Amaral, L Glass, JM Hausdorff, PCh Ivanov, RG Mark, et al., "PhysioBank PhysioToolkit and PhysioNet: Components of a New Research Re-source for Complex Physiologic Signals", Circulation, vol. 101, no. 23, pp. e215-e220, June 2000.

[5] M. U. Khan, S. Aziz, K. Iqtidar, A. Saud and Z. Azhar, "Biometric Authentication System Based on Electrocardiogram (ECG)", 2019 13th International Conference on Mathematics Actuarial Science Computer Science and Statistics (MACS), pp. 1-6, 2019.

[6] M. U. Khan, S. Aziz, K. Iqtidar, A. Saud and Z. Azhar, "Biometric Authentication System Based on Electrocardiogram (ECG)," 2019 13th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS), Karachi, Pakistan, 2019, pp. 1-6, doi: 10.1109/MACS48846.2019.9024820.

[7] N. ALI DEEB, A. DAHER, M. CHAKOUCH, S. NACHAR and W. KAMALI, "Development of a New Biometric Authentication Approach Based on Electrocardiogram Signals," 2019 Fifth International Conference on Advances in Biomedical Engineering (ICABME), Tripoli, Lebanon, 2019, pp. 1-4, doi: 10.1109/ICABME47164.2019.8940208.

[8] A. K and G. N. K. Murthy, "Data Acquisition of Iris and ECG for Person Biometric Authentication and Verification," 2023 International Conference on Smart Systems for applications in Electrical Sciences (ICSSES), Tumakuru, India, 2023, pp. 1-6, doi: 10.1109/ICSSES58299.2023.10200698.

[9] T. Repcik et al., "Biometric Authentication Using the Unique Characteristics of the ECG Signal," 2020 Computing in Cardiology, Rimini, Italy, 2020, pp. 1-4, doi: 10.22489/CinC.2020.321.

[10] R. R. Thirrunavukkarasu, R. Likkitha, V. C. Kaviya and S. Madhumathi, "Fetal ECG denoising using Adaptive Thresholding and Extended kalman filter With Dynamic Time Warping," 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2020, pp. 649-654, doi: 10.1109/ICACC

[11] Lugovaya T.S. Biometric human identification based on electrocardiogram. [Master's thesis] Faculty of Computing Technologies and Informatics, Electrotechnical University "LETI", Saint-Petersburg, Russian Federation; June 2005.

Appendix:

```
import numpy as np
from scipy.signal import butter, filtfilt
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
import pywt
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import StratifiedKFold
from fastdtw import fastdtw
from scipy.signal import find_peaks



def bandpass_filter(data, lowcut, highcut, fs, order=4):
    nyquist = 0.5 * fs
    low = lowcut / nyquist
    high = highcut / nyquist
    b, a = butter(order, [low, high], btype='band')

    # Ensure that the length of the input vector is greater than the filter order
    if len(data) <= order:
        raise ValueError("The length of the input vector must be greater than the filter order.")

    filtered_data = filtfilt(b, a, data)
    return filtered_data

# Example usage:
# Replace 'your_ecg_data.csv' with the path to your CSV file and 'your_sampling_rate' with the sampling rate of your data.
csv_file_path = 'rec_30_2.csv'  # Replace this with the path to your CSV file
csv_file_path_y = 'rec_70_1.csv'
csv_file_path_y1 = 'rec_60_1.csv'
csv_file_path_y2 = 'rec_30_2.csv'
csv_file_path_y3 = 'rec_40_1.csv'
csv_file_path_y4 = 'rec_10_1.csv'
your_sampling_rate = 1000  # Replace this with your actual sampling rate

# Load ECG data from CSV
ecg_data = pd.read_csv(csv_file_path).iloc[0:,0].values  # Assuming 'ECG' is the column name in your CSV file
ecg_data_y = pd.read_csv(csv_file_path_y).iloc[0:,0].values
ecg_data_y1 = pd.read_csv(csv_file_path_y1).iloc[0:,0].values
ecg_data_y2 = pd.read_csv(csv_file_path_y2).iloc[0:,0].values
ecg_data_y3 = pd.read_csv(csv_file_path_y3).iloc[0:,0].values
ecg_data_y4 = pd.read_csv(csv_file_path_y4).iloc[0:,0].values



lowcut = 2.0  # Lower cutoff frequency in Hz
highcut = 50.0  # Upper cutoff frequency in Hz

filtered_ecg = bandpass_filter(ecg_data, lowcut, highcut, your_sampling_rate)
filtered_ecg_y = bandpass_filter(ecg_data_y, lowcut, highcut, your_sampling_rate)
filtered_ecg_y1 = bandpass_filter(ecg_data_y1, lowcut, highcut, your_sampling_rate)
filtered_ecg_y2 = bandpass_filter(ecg_data_y2, lowcut, highcut, your_sampling_rate)
filtered_ecg_y3 = bandpass_filter(ecg_data_y3, lowcut, highcut, your_sampling_rate)
filtered_ecg_y4 = bandpass_filter(ecg_data_y4, lowcut, highcut, your_sampling_rate)

ecg_csv = [ecg_data, ecg_data_y,ecg_data_y1,ecg_data_y2,ecg_data_y3,ecg_data_y4]
filtered_signals = [filtered_ecg, filtered_ecg_y, filtered_ecg_y1, filtered_ecg_y2,filtered_ecg_y3,filtered_ecg_y4]

for i in range(len(ecg_csv)):
    # Plot the original and filtered signals in separate subplots
    plt.figure(figsize=(12, 8))

    # Plot original ECG signal
    plt.subplot(2, 1, 1)
    plt.plot(ecg_csv[i], label='Original ECG')
    plt.title('Original ECG Signal')
    plt.xlabel('Sample')
    plt.ylabel('Amplitude')
    plt.legend()

    # Plot filtered ECG signal
    plt.subplot(2, 1, 2)
    plt.plot(filtered_signals[i], label='Filtered ECG (2-50 Hz)')
    plt.title('Filtered ECG Signal (2-50 Hz)')
    plt.xlabel('Sample')
    plt.ylabel('Amplitude')
    plt.legend()

    plt.tight_layout()  # Adjust layout to prevent overlap
    plt.show()



def detect_all_peaks(ecg_signal, threshold_factor=0.62):
    # Find peaks in the ECG signal using adaptive thresholding
    peaks, _ = find_peaks(ecg_signal, height=np.max(ecg_signal) * threshold_factor)

    return peaks

# Example usage:
# Replace 'filtered_ecg_signal' with your actual filtered ECG signal

for i in range(len(ecg_csv)):
    detected_peaks = detect_all_peaks((filtered_signals[i])[:3000])

    # Plot the ECG signal with all detected peaks
    plt.figure(figsize=(14, 5))
    plt.plot(filtered_signals[i][:3000], label='Filtered ECG Signal')
    plt.plot(detected_peaks, filtered_signals[i][:3000][detected_peaks], 'go', label='Detected Peaks')
    plt.title('Detection of All Peaks in Filtered ECG Signal')
    plt.xlabel('Sample Index')
```

```
    plt.ylabel('Amplitude')
    plt.legend()
    plt.show()


# Simulate an ECG signal (replace this with your actual signal)
fs = 1000  # Sampling frequency



def wavelet_feature_extraction(ecg_signal, wavelet='sym4', level=4):
    # Perform wavelet decomposition
    coeffs = pywt.wavedec(ecg_signal, wavelet, level=level)

    # Plot the original signal and wavelet coefficients
    plt.figure(figsize=(13, 10))
    plt.plot(ecg_signal)
    plt.title('Original ECG Signal')
    plt.xlabel('Time (s)')
    plt.ylabel('Amplitude')
    plt.show()



    plt.figure(figsize=(20, 18))
    plt.subplot(6,1,1)
    plt.plot(coeffs[0])
    plt.title(f'Level {1} Coefficients')
    plt.xlabel('Coefficient Index')
    plt.ylabel('Amplitude')

    plt.subplot(6,1,3)
    plt.plot(coeffs[1])
    plt.title(f'Level {2} Coefficients')
    plt.xlabel('Coefficient Index')
    plt.ylabel('Amplitude')

    plt.subplot(6,1,6)
    plt.plot(coeffs[2])
    plt.title(f'Level {3} Coefficients')
    plt.xlabel('Coefficient Index')
    plt.ylabel('Amplitude')

    plt.figure(figsize=(20, 18))
    plt.subplot(5,1,1)
    plt.plot(coeffs[3])
    plt.title(f'Level {4} Coefficients')
    plt.xlabel('Coefficient Index')
    plt.ylabel('Amplitude')

    plt.subplot(5,1,3)
    plt.plot(coeffs[4])
    plt.title(f'Level {5} Coefficients')
    plt.xlabel('Coefficient Index')
    plt.ylabel('Amplitude')

    plt.show()
    # Extract statistical features from wavelet coefficients
    feature_vector = np.hstack([np.mean(c) for c in coeffs])

    # Display the extracted features
    print("Extracted Features:")
    print(feature_vector)

    return feature_vector

# Example usage:
# Replace ecg_signal with your actual ECG signal
 # Replace this with your actual ECG signal
feature_vector = wavelet_feature_extraction(filtered_ecg)
feature_vector_y = wavelet_feature_extraction(filtered_ecg_y)
feature_vector_y1 = wavelet_feature_extraction(filtered_ecg_y1)
feature_vector_y2 = wavelet_feature_extraction(filtered_ecg_y2)
feature_vector_y3 = wavelet_feature_extraction(filtered_ecg_y3)
feature_vector_y4 = wavelet_feature_extraction(filtered_ecg_y4)


feature_vector_array = [feature_vector, feature_vector_y, feature_vector_y1, feature_vector_y2, feature_vector_y3, feature_vector_y4]



def calculate_dtw_distance(template_feature_vector, query_feature_vector):
    distance, path = fastdtw(template_feature_vector, query_feature_vector)
    return distance, path

# Example usage:
# Replace template_feature_vector and query_feature_vector with your actual feature vectors
template_feature_vector = feature_vector
query_feature_vector = filtered_signals
dtw_distance_array = []
alignment_path_array = []
# Calculate DTW distance and path

dtw_distance, alignment_path = calculate_dtw_distance(template_feature_vector, filtered_ecg_y)
print(f"DTW Distance: {dtw_distance}")
dtw_distance_array.append(dtw_distance)
alignment_path_array.append(alignment_path)

# Plot the alignment matrix heatmap
plt.figure(figsize=(8, 6))
plt.imshow(np.array(alignment_path).T, origin='lower', cmap='viridis', interpolation='nearest')
plt.title('DTW Alignment Matrix')
plt.xlabel('Template Feature Vector Index')
plt.ylabel('Query Feature Vector Index')
plt.colorbar(label='Alignment Cost')
plt.show()

# Display the DTW distance
```

```
genuine_scores = feature_vector
imposter_scores = feature_vector_array



def calculate_roc(genuine_scores, imposter_scores):
    # Combine genuine and imposter scores and labels
    all_scores = np.concatenate([genuine_scores, imposter_scores])
    true_labels = np.concatenate([np.ones_like(genuine_scores), np.zeros_like(imposter_scores)])

    # Compute ROC curve and area under the curve (AUC)
    fpr, tpr, thresholds = roc_curve(true_labels, -all_scores)  # Using -scores because roc_curve considers higher values as better
    roc_auc = auc(fpr, tpr)

    # Plot ROC curve
    plt.figure(figsize=(8, 8))
    plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'AUC = {roc_auc:.2f}')
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic')
    plt.legend(loc="lower right")
    plt.show()

    # Find the threshold that gives a balanced error rate (equal FPR and FNR)
    eer_threshold = thresholds[np.argmin(np.abs(fpr - (1 - tpr)))]
    print(f"Equal Error Rate (EER) Threshold: {eer_threshold}")

    return eer_threshold


threshold_array = []
# Example usage:
# Replace genuine_scores and imposter_scores with your actual score arrays
for i in range(1, len(ecg_csv)):
    threshold = calculate_roc(genuine_scores, feature_vector_y)
    print(f"Chosen Threshold{i}: {threshold}")
    threshold_array.append(threshold)




def authenticate_claimed_identity(XTI, Xq, threshold, names):
    distance = calculate_dtw_distance(XTI, Xq)[0]

    print("Distance", distance)

    if distance <= threshold:
        print(f"{names} Identity accepted as a genuine user.")
    else:
        print(f"{names} Identity rejected. Considered an imposter.")

# Example usage:
# Replace these with your actual feature vectors and threshold
XTI = feature_vector
Xq = feature_vector_array
names = ['ecg0', 'ecg1', 'ecg2', 'ecg3', 'ecg4', 'ecg5']


for i in range(1, len(ecg_csv)):
    # Authenticate the claimed identity
    authenticate_claimed_identity(XTI, Xq[i], threshold,names[i])



def evaluate_metrics(genuine_scores, imposter_scores):
    # Combine genuine and imposter scores
    all_scores = np.concatenate((genuine_scores, imposter_scores))

    # Create labels for genuine (1) and imposter (0) scores
    labels = np.concatenate((np.ones_like(genuine_scores), np.zeros_like(imposter_scores)))

    # Calculate ROC curve and area under the curve (AUC)
    fpr, tpr, _ = roc_curve(labels, all_scores)
    roc_auc = auc(fpr, tpr)

    # Calculate equal error rate (EER)
    eer = np.abs(fpr[np.argmax(tpr) - 1] - (1 - tpr[np.argmax(tpr) - 1]))

    # Calculate false accept rate (FAR), false reject rate (FRR), and accuracy
    distance = (calculate_dtw_distance(XTI, imposter_scores))[0]
    far = fpr[np.argmax(tpr) - 1]
    if distance <= threshold:
        frr = 0
    else:
        frr = 1 - tpr[np.argmax(tpr) - 1]


    return roc_auc, eer, far, frr

print("Imposter_scores ",imposter_scores)

for i in range(1, len(ecg_csv)):
    roc_auc, eer, far, frr = evaluate_metrics(genuine_scores, imposter_scores[i])
    print(f"Resut for ECG Signal {i} \n")
    # Display the results
    print("ROC AUC:", roc_auc)
    print("Equal Error Rate (EER):", eer)
    print("False Accept Rate (FAR):", far)
    print("False Reject Rate (FRR):", frr)
    print("\n")

def calculate_accuracy(decisions, ground_truth):
    # Ensure decisions and ground_truth have the same length
    if len(decisions) != len(ground_truth):
        raise ValueError("Decisions and ground_truth must have the same length.")
```

```
    # Calculate accuracy
    correct_predictions = np.sum(decisions == ground_truth)
    total_samples = len(ground_truth)
    accuracy = (correct_predictions / total_samples)*2

    return accuracy

# Example usage:
# Replace decisions and ground_truth with your actual decisions and ground truth
for i in range(1, len(ecg_csv)):
    ground_truth = np.concatenate([np.ones_like(genuine_scores), np.zeros_like(imposter_scores[i])])
    all_scores = np.concatenate([genuine_scores, imposter_scores[i]])
    decisions = (all_scores >= threshold).astype(int)  # Using the threshold determined from ROC analysis

    accuracy = calculate_accuracy(decisions, ground_truth)
    print(f"Accuracy for ECG Signal {i}:", accuracy*100, "%")
```