IoT Trust and Anomaly Detection Framework using ISAC Simulation and Lightweight Machine Learning

Submitted by: Amna Siddiqui

# Task 2.1: Framework Design and Simulation

The simulation includes a fusion center, edge nodes, and IoT devices connected in a topology. It helps illustrate the ISAC communication structure.

Each IoT device is also assigned physical-layer values: RSSI, CSI, and ToF.

Key Code Snippet:

import networkx as nx  
import matplotlib.pyplot as plt  
import random  
import hashlib  
import time  
  
G = nx.Graph()  
G.add\_node("FusionCenter", type="fusion")  
G.add\_node("Edge1", type="edge")  
G.add\_node("Edge2", type="edge")  
for i in range(5):  
 G.add\_node(f"IoT{i+1}", type="iot")  
G.add\_edges\_from([...])

for node in G.nodes:  
 if G.nodes[node]['type'] == 'iot':  
 G.nodes[node]['RSSI'] = random.uniform(-90, -30)  
 G.nodes[node]['CSI'] = random.uniform(0.1, 1.0)  
 G.nodes[node]['ToF'] = random.uniform(1.0, 5.0)

# Task 2.2: Algorithm Implementation

Trust score is calculated using RSSI, CSI, and ToF. Feedback is checked for integrity using hashing.

Main logic for scoring and anomaly detection:

def compute\_ascore(node):  
 w = {'RSSI': 0.3, 'CSI': 0.4, 'ToF': 0.3}  
 return (w['RSSI'] \* abs(node['RSSI']) + w['CSI'] \* node['CSI'] + w['ToF'] \* node['ToF'])  
  
for node in G.nodes:  
 if G.nodes[node]['type'] == 'iot':  
 score = compute\_ascore(G.nodes[node])  
 G.nodes[node]['Ascore'] = score  
 G.nodes[node]['trusted'] = 30 < score < 90

def hash\_feedback(data, nonce, key):  
 return hashlib.sha256(f"{data}{nonce}{key}".encode()).hexdigest()  
  
for node in G.nodes:  
 if G.nodes[node]['type'] == 'iot':  
 A\_iot = sum([random.randint(1, 5) for \_ in range(10)])  
 A\_fusion = A\_iot if random.random() > 0.3 else A\_iot + 7  
 fv = hash\_feedback(A\_iot, nonce, private\_key)  
 fs = hash\_feedback(A\_fusion, nonce, private\_key)  
 G.nodes[node]['anomaly'] = fv != fs

# Task 2.3: Integration and Testing

Exporting CSVs and measuring trust and anomaly data for multiple simulation runs. Also includes latency simulation.

CSV + Output Code:

import pandas as pd  
iot\_data = []  
for node in G.nodes:  
 if G.nodes[node]['type'] == 'iot':  
 iot\_data.append({...})  
df = pd.DataFrame(iot\_data)  
df.to\_csv("iot\_device\_status\_report.csv", index=False)

# Task 2.4: Field Deployment and Refinement

Though field hardware testing wasn’t done, the code is ready to adapt. The model was refined iteratively using log data.

Simulation latency example:

start = time.time()  
time.sleep(random.uniform(0.1, 0.3))  
end = time.time()  
print(f"Latency: {end - start:.3f} seconds")