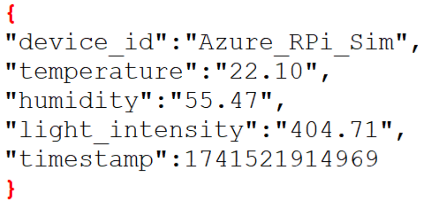
STEP-BY-STEP EXPERIMENTAL PROCEDURE

of

“Securing real-time IoT systems with blockchain-enabled edge computing and anomaly detection”

1. IoT Sensor Setup and Data Verification

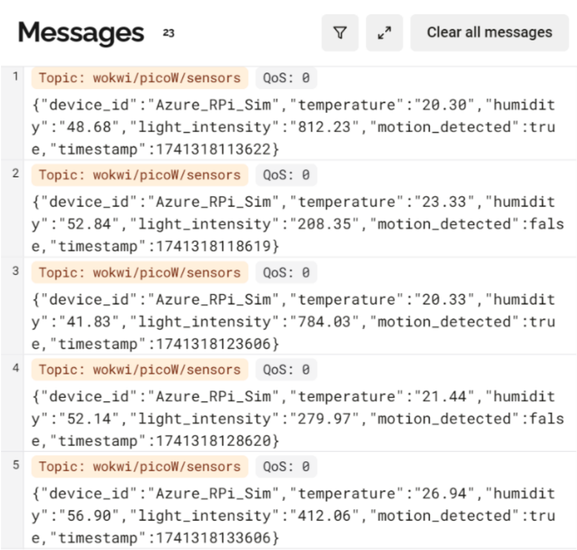
- Connect the DHT11 and LDR sensors to Raspberry Pi Pico W GPIO pins.
- Deploy JavaScript to collect sensor readings at fixed intervals of 5 seconds. In this experiment, random data is generated with a calculated threshold.
- Run JavaScript to check the data transmission in the simulator terminal. The data will return in JSON format as shown in Fig. 1.



```
{
"device_id":"Azure_RPi_Sim",
"temperature":"22.10",
"humidity":"55.47",
"light_intensity":"404.71",
"timestamp":1741521914969
}
```

Figure 1: JSON Data

- Configure MQTT client through JavaScript with HiveMQ provided constant BROKER_URL, MQTT_PORT, CLIENT_ID, USERNAME, and PASSWORD.
- Connect to the HiveMQ broker with TLS encryption enabled.
- Define a topic to publish sensor data.
- Run the JavaScript and validate if the generated data in the terminal is the same as that in HiveMQ (Fig. 2).



```
1 Topic: wokwi/picoW/sensors  QoS: 0
{"device_id":"Azure_RPi_Sim","temperature":"20.30","humidit
y":"48.68","light_intensity":"812.23","motion_detected":tru
e,"timestamp":1741318113622}
2 Topic: wokwi/picoW/sensors  QoS: 0
{"device_id":"Azure_RPi_Sim","temperature":"23.33","humidit
y":"52.84","light_intensity":"208.35","motion_detected":fals
e,"timestamp":1741318118619}
3 Topic: wokwi/picoW/sensors  QoS: 0
{"device_id":"Azure_RPi_Sim","temperature":"20.33","humidit
y":"41.83","light_intensity":"784.03","motion_detected":tru
e,"timestamp":1741318123606}
4 Topic: wokwi/picoW/sensors  QoS: 0
{"device_id":"Azure_RPi_Sim","temperature":"21.44","humidit
y":"52.14","light_intensity":"279.97","motion_detected":fals
e,"timestamp":1741318128620}
5 Topic: wokwi/picoW/sensors  QoS: 0
{"device_id":"Azure_RPi_Sim","temperature":"26.94","humidit
y":"56.90","light_intensity":"412.06","motion_detected":tru
e,"timestamp":1741318133606}
```

Figure 2: HiveMQ Message Received from Sensors

2. Anomaly Detection Algorithm

The algorithm is deployed in Python, which receives data in the edge node where the anomaly scan is done, and data is filtered before transferring to the blockchain layer.

- Computes rolling mean and standard deviation and removes NaN values from rolling computations.
- Normalizes data using StandardScaler library from Sklearn.

- Uses Isolation Forest to find outliers and labels anomalies and counts them.
- Accuracy analysis (F1 score) and confusion matrix is required to implement so that algorithm performance is monitored and modified as required.

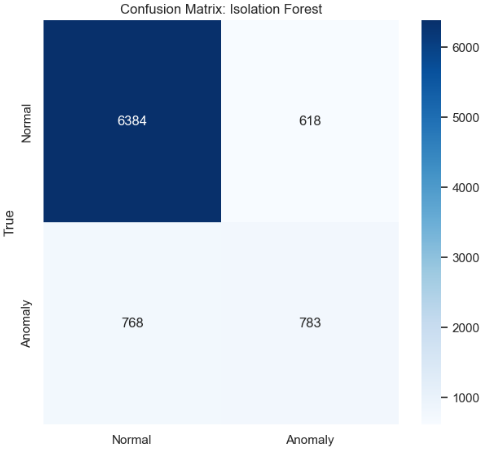
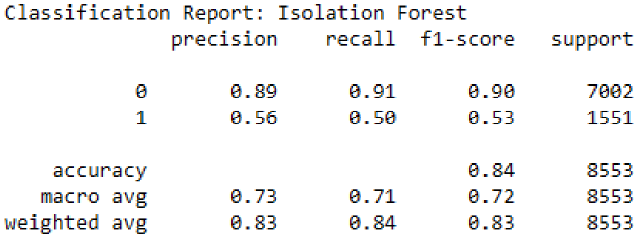


Figure 3: Confusion Matrix

The confusion matrix in Fig. 3 demonstrates the difference in True Positives (TP), False Positives (FP), False Negatives (FN), and True Negatives (TN).

Predicted



```
Classification Report: Isolation Forest
              precision    recall  f1-score   support

           0       0.89      0.91      0.90      7002
           1       0.56      0.50      0.53      1551

    accuracy                           0.84      8553
   macro avg       0.73      0.71      0.72      8553
weighted avg       0.83      0.84      0.83      8553
```

Figure 4: Classification Report

Fig. 4 demonstrates the accuracy of the anomaly detection using precision, recall, and F1-score.

- Plot visualization for quick review of anomaly.

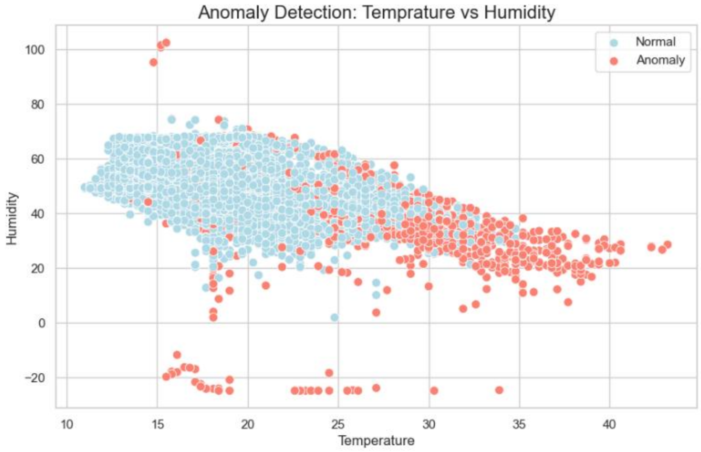


Figure 5: Temperature vs Humidity

Fig. 5 demonstrates the anomaly detected in the "salmon" color, indicating inconsistent data points.

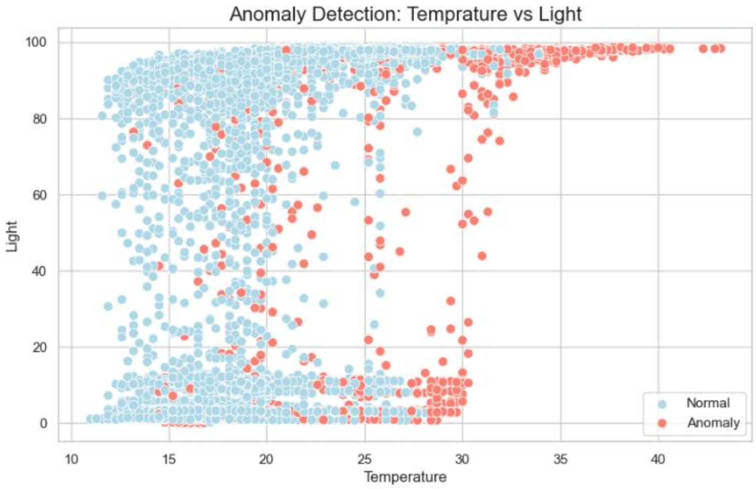


Figure 6: Temperature vs Light

Similarly, Fig. 6 demonstrates the anomaly detection for light vs temperature.

- Function creation for periodic real-time data streaming is required so that data trends can be monitored, and the algorithm can be tuned.

3. Latency Measurement

- Configure latency measurement so that the device (publisher) sends a message to the MQTT broker with a timestamp.
- The MQTT broker receives the message, processes it, and potentially forwards it to subscribers.
- The subscriber receives the message, extracts the timestamp, and calculates the latency.

Latency details are provided in Table 1.

Measurement Type	Timestamp	Device/Node	Latency (ms)	Notes
MQTT Publish Latency	10:00:01	IoT Layer	350	Time taken for the sensor data to be published to HiveMQ Cloud. Initial readings show moderate latency due to normal network traffic.
Blockchain Transaction Latency	10:00:10	IoT Layer	4200	Time taken to confirm the blockchain transaction on Ethereum Sepolia Testnet. Higher latency is observed due to blockchain consensus delays.

Edge Processing Speed	10:00:12	Edge Layer	120	Time to detect anomalies and process data at the edge node. Processing remains efficient within the expected range.
MQTT Publish Latency	10:05:05	IoT Layer	310	Slightly faster due to network optimization. Possible improvement in broker response time.
Blockchain Transaction Latency	10:05:20	IoT Layer	4300	Slight increase due to network congestion. Potential increase in pending transactions on the Ethereum network.
Edge Processing Speed	10:05:22	Edge Layer	115	Stable edge processing time. No significant delays were observed in anomaly detection and data handling.
MQTT Publish Latency	10:10:05	IoT Layer	340	Average latency for data to reach MQTT broker. Possible minor fluctuations in network performance.
Blockchain Transaction Latency	10:10:15	IoT Layer	4150	Regular blockchain transaction confirmation time. No major network congestion at this timestamp.
Edge Processing Speed	10:10:18	Edge Layer	110	Efficient processing at edge node for anomaly detection. Slight improvement compared to previous readings.
MQTT Publish Latency	10:15:05	IoT Layer	300	Network optimized for lower latency during continuous data send. Indicates improved broker response time.
Blockchain Transaction Latency	10:15:18	IoT Layer	4000	Stable transaction time despite slight network delays. Represents a slight decrease in latency compared to earlier values.
Edge Processing Speed	10:15:20	Edge Layer	125	Minor increase due to additional edge processing tasks. Could be due to a higher volume of incoming data for analysis.

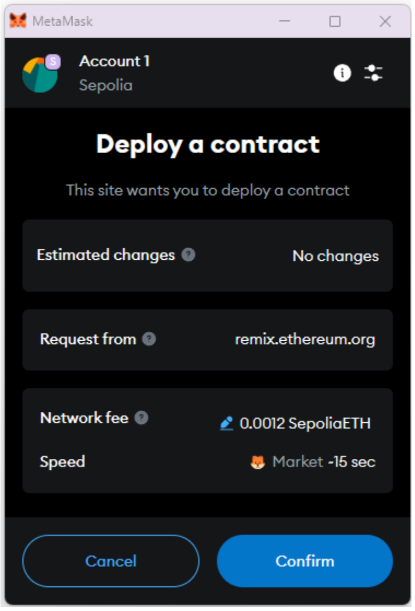
Table 1:Latency and Data Processing Time

4. Blockchain Storage and Verification

- Creating a Contract and Deployment:
  - Create a new file.
  - Code a contract in the file (IoTData.sol), the function storeData() to record sensor data immutably.
  - Compile the contract Deploy a smart contract on the Ethereum Sepolia Testnet. Execute transactions via MetaMask to store sensor readings on-chain.

- ○ Verify data storage by retrieving transaction details from the blockchain explorer.

- ○



Account 1
Sepolia

Deploy a contract

This site wants you to deploy a contract

Estimated changes: No changes

Request from: remix.ethereum.org

Network fee: 0.0012 SepoliaETH

Speed: Market -15 sec

5. User Interaction and Security Measures

- Users authenticate via MetaMask to access stored sensor data.
- Web application fetches blockchain-stored values and compares them to MQTT data.
- Implement role-based access control within smart contracts to ensure only verified users can retrieve sensitive data.