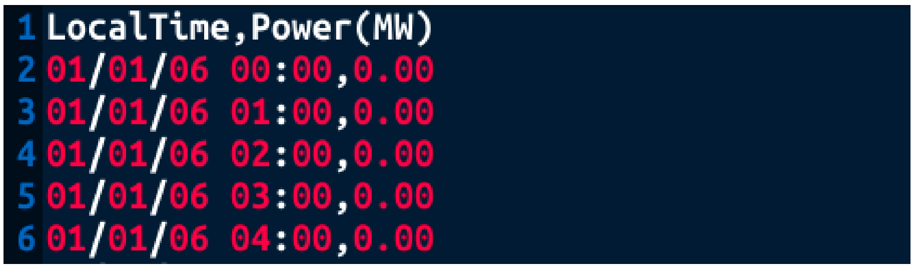
EKF vs UKF Implementation

We have implemented two types of Kalman filter - the **Extended Kalman Filter (EKF)** and the **Unscented Kalman Filter (UKF)** to predict power values using both synthetic data and real-world solar data.

Initially, we tested the Kalman filters on synthetic data to validate their functionality. The synthetic data used here consists of a simple time-series of power measurements, where we simulate the power signal over time and compare the predicted estimates from both EKF and UKF with the real measurements. This synthetic dataset was designed to represent a simple system with predictable noise characteristics, making it ideal for understanding the core behaviour of the Kalman filters.

Thereafter, we used real-world solar power data from the National Renewable Energy Laboratory (NREL), which provides time-series data of solar power measurements. The specific dataset used is from a 9 MW solar power plant, covering an entire year of hourly data.

- **Data Format**: The dataset consists of two primary columns:
  - **LocalTime**: The timestamp of the power measurement in the format "MM/DD/YY HH".
  - **Power(MW)**: The measured power output in megawatts (MW).

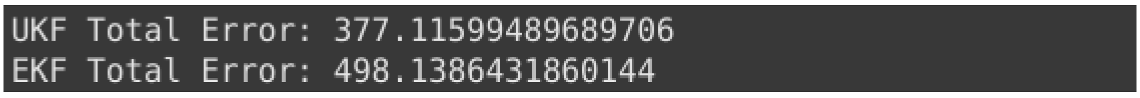


```
LocalTime,Power(MW)
01/01/06 00:00,0.00
01/01/06 01:00,0.00
01/01/06 02:00,0.00
01/01/06 03:00,0.00
01/01/06 04:00,0.00
```

- Both filters were implemented using basic time-series models, assuming that the true power signal follows a constant velocity model. The process noise (Q) and measurement noise (R) were tuned according to typical settings in Kalman filtering for power prediction tasks. The performance of both filters was evaluated by calculating the Mean Squared Error (MSE) of the estimates and comparing the errors of EKF and UKF.

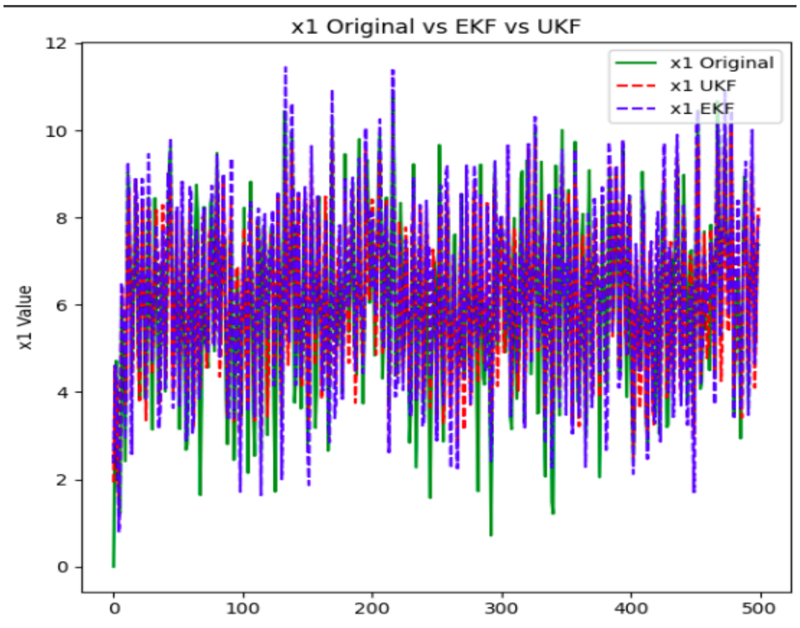
Results: Error Comparison and Plot Analysis

- **Error Comparison (MSE)**: The Mean Squared Error (MSE) was calculated for both the EKF and UKF to compare their prediction accuracies. The EKF, being a linear filter, struggles with nonlinearities, while the UKF, with its ability to handle nonlinearities more effectively, should ideally produce smaller errors. The results showed that the **UKF consistently produced a lower error compared to EKF**.



```
UKF Total Error: 377.11599489689706
EKF Total Error: 498.1386431860144
```

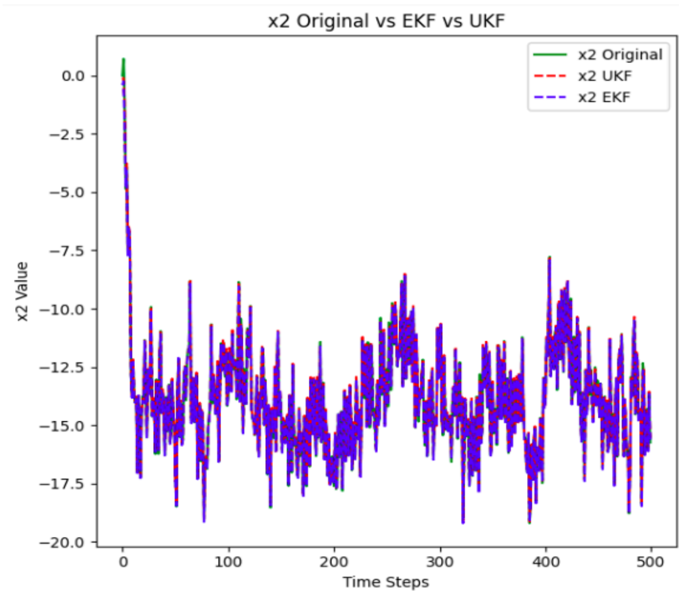
- **Plot Comparison**: To visually compare the performance of the filters, the predicted values from both EKF and UKF were plotted against the real solar power measurements over time. Confidence ellipses were also plotted to provide a sense of the uncertainty in the state estimates.

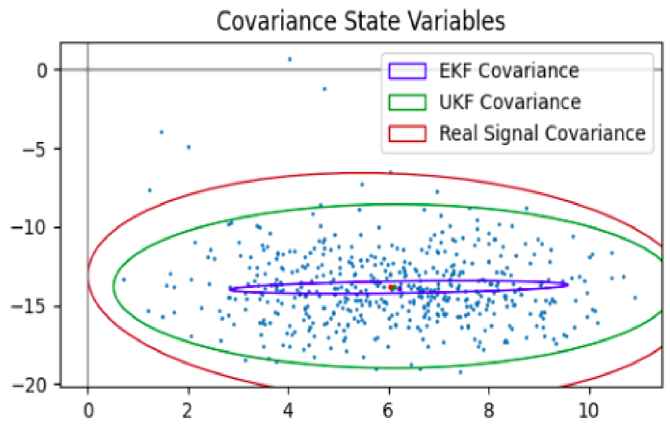


EKF Plot: Noticeable deviations from the actual values due to the linear assumption in the model.

UKF Plot: It closely tracks the real data, with fewer fluctuations and better handling of the nonlinearities in the data.

The UKF's confidence ellipses are narrower and more aligned with the actual data.





Testing with different Hyperparameter sets

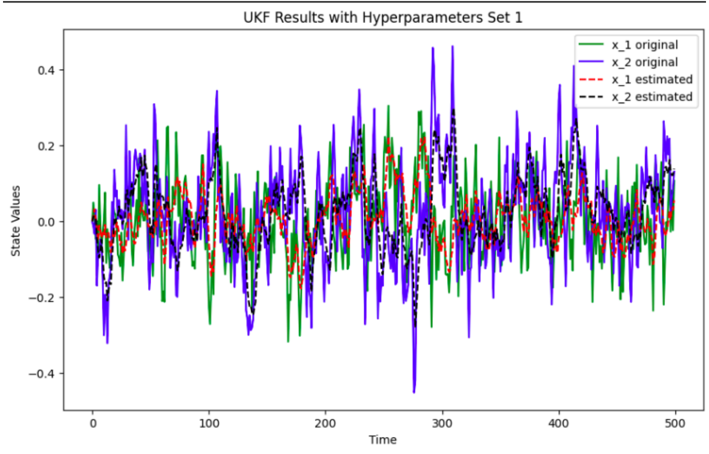
We have also evaluated the performance of the UKF technique by testing it with different sets of hyperparameters. The key parameters we adjust are:

- **Initial Covariance (init_cov)**: This sets our initial uncertainty about the state.
- **Process Noise (process_noise)**: We use this to model the uncertainty in the system's dynamics.
- **Measurement Noise (meas_noise)**: This accounts for noise in the measurements we receive.

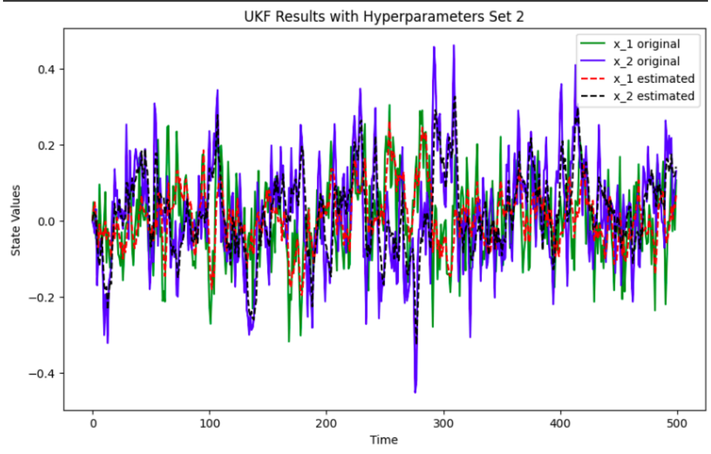
For each hyperparameter set, we generate a plot showing:

- The **original states** (x_1 and x_2) in green and blue.
- The **estimated states** (x_1 and x_2) in red dashed and black dashed lines.

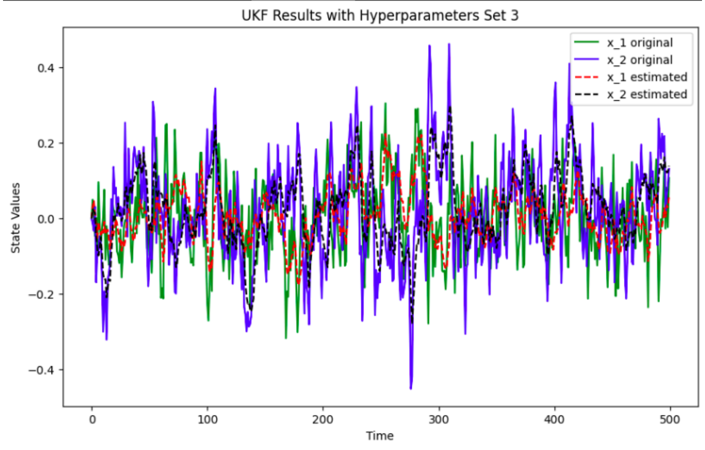
Hyperparameter set 1 (0.1, 0.1, 0.1)



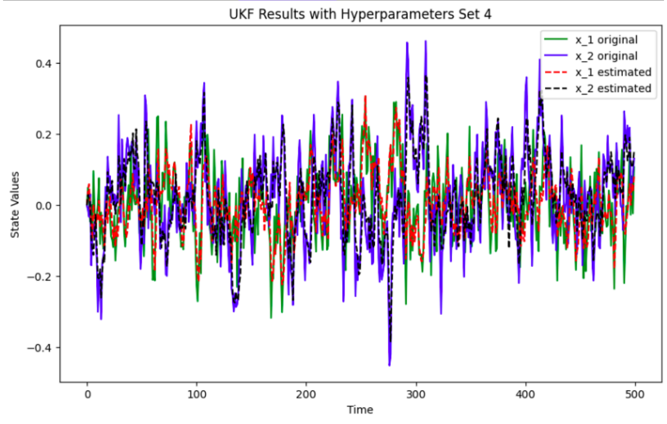
Hyperparameter set 2 (0.5, 0.2, 0.1)



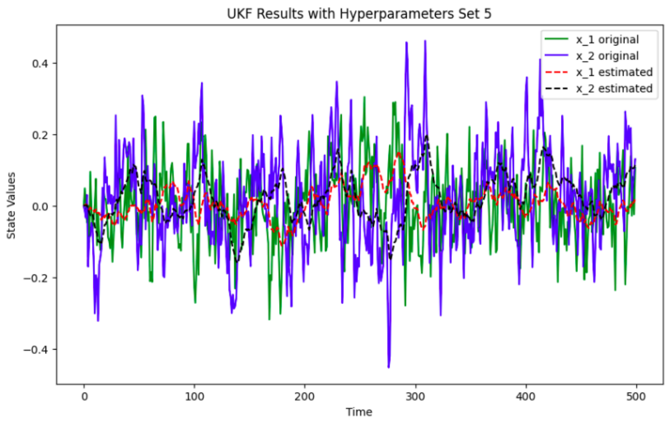
Hyperparameter set 3 (1.0, 0.1, 0.5)



Hyperparameter set 4 (0.1, 0.5, 0.5)



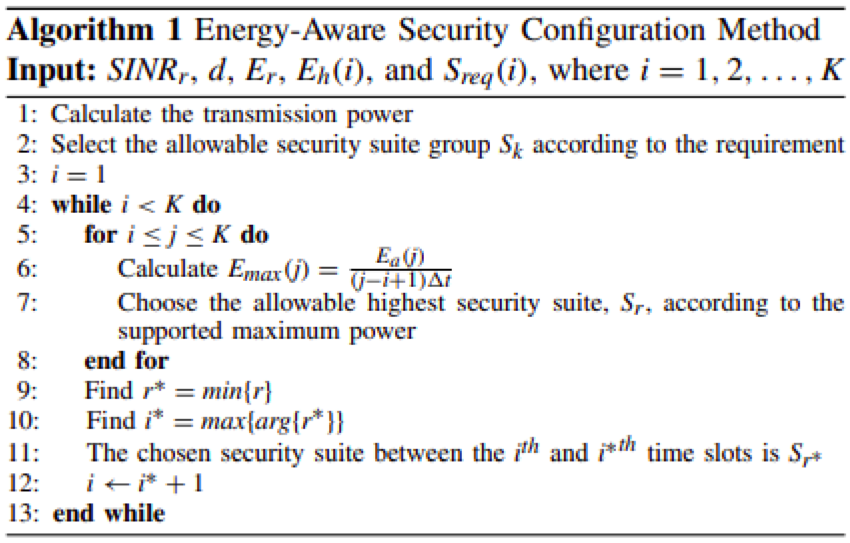
Hyperparameter set 5 (0.5, 0.2, 0.1)



Energy Aware Security Configuration

Original Algorithm

The original energy aware security configuration algorithm is designed to allocate the **highest possible security suite** for user equipment (UE) while taking energy constraints into account. For each UE, the algorithm calculates the available energy during a given time interval by combining the **residual energy** and the **harvested energy**. It then determines the **maximum allowable energy (Emax)** that can be used during the interval, based on the energy available and the duration of the interval. From the set of allowable security suites, the algorithm selects the **highest security level** that meets the **Security Requirement (Sreq)** of the UE. The process iterates over all time intervals, ensuring that energy resources are utilized efficiently while fulfilling the required security levels.



Algorithm 1 Energy-Aware Security Configuration Method

Input: $SINR_r$, d, E_r, $E_h(i)$, and $S_{req}(i)$, where $i = 1, 2, \dots, K$

```
1: Calculate the transmission power
2: Select the allowable security suite group $S_k$ according to the requirement
3: $i = 1$
4: while $i < K$ do
5:     for $i \leq j \leq K$ do
6:         Calculate $E_{max}(j) = \frac{E_a(j)}{(j-i+1)\Delta t}$
7:         Choose the allowable highest security suite, $S_r$, according to the
           supported maximum power
8:     end for
9:     Find $r* = min\{r\}$
10:    Find $i* = max\{arg\{r*\}\}$
11:    The chosen security suite between the $i^{th}$ and $i*^{th}$ time slots is $S_{r*}$
12:    $i \leftarrow i* + 1$
13: end while
```

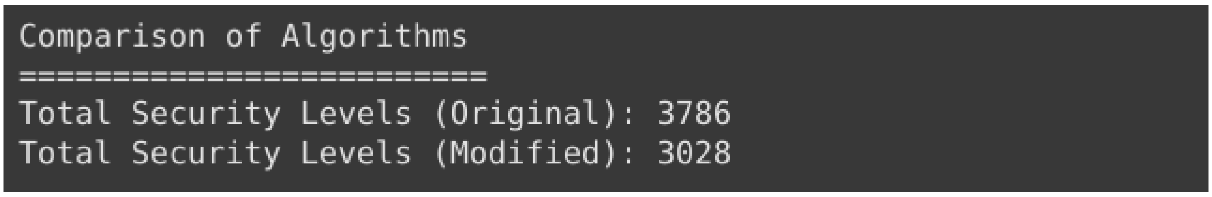
Proposed Algorithm

The modified algorithm builds on the original by introducing a **Security Threshold (S_{th})** for each UE, which represents the **maximum practical security requirement** for the UE. This ensures that the security level assigned does not exceed the threshold, even if the energy allows for a higher level. Similar to the original algorithm, it calculates the **residual energy**, **harvested energy**, and **maximum allowable energy (E_{max})** for each interval. However, when determining the security suite, the **Security Requirement (S_{req})** is compared with the **Security Threshold (S_{th})** and the lower of the two is selected. This guarantees that the assigned security suite meets the minimum requirement (S_{req}) while avoiding unnecessary overprovisioning of security.

By incorporating the **S_{th}**, the modified algorithm optimizes energy consumption more effectively. It iterates over all intervals, balancing energy constraints and practical security needs, thereby preventing the allocation of excessive security levels.

Improvements and Comparison

- **Reduced Energy Consumption:** The modified algorithm avoids assigning unnecessarily high-security levels by capping the selection at the Security Threshold (S_{th}). This prevents overprovisioning and conserves energy, allowing the UE to operate longer on the available energy.
- **Adaptability to Practical Needs**: By incorporating the S_{th}, the modified algorithm accounts for the UE's practical security requirements. This ensures that energy is not wasted on achieving security levels that exceed the UE's actual needs.
- **Prevention of Over-Securing:** The original algorithm might allocate unnecessarily high security levels due to energy availability, which could lead to higher processing and transmission overhead. The modified algorithm avoids this by adhering to the Sth.



```
Comparison of Algorithms
========================
Total Security Levels (Original): 3786
Total Security Levels (Modified): 3028
```

The comparison demonstrates the effectiveness of the modified algorithm in reducing overall security levels while maintaining necessary security requirements. The original algorithm resulted in a total of 3786 security levels across all intervals, whereas the modified algorithm, incorporating the maximum security threshold (S_{th}), reduced this to 3028. This reduction highlights the modified algorithm's ability to conserve energy by avoiding excessive security provisioning, ensuring efficient resource utilization while adhering to practical security needs.

Dataset Used

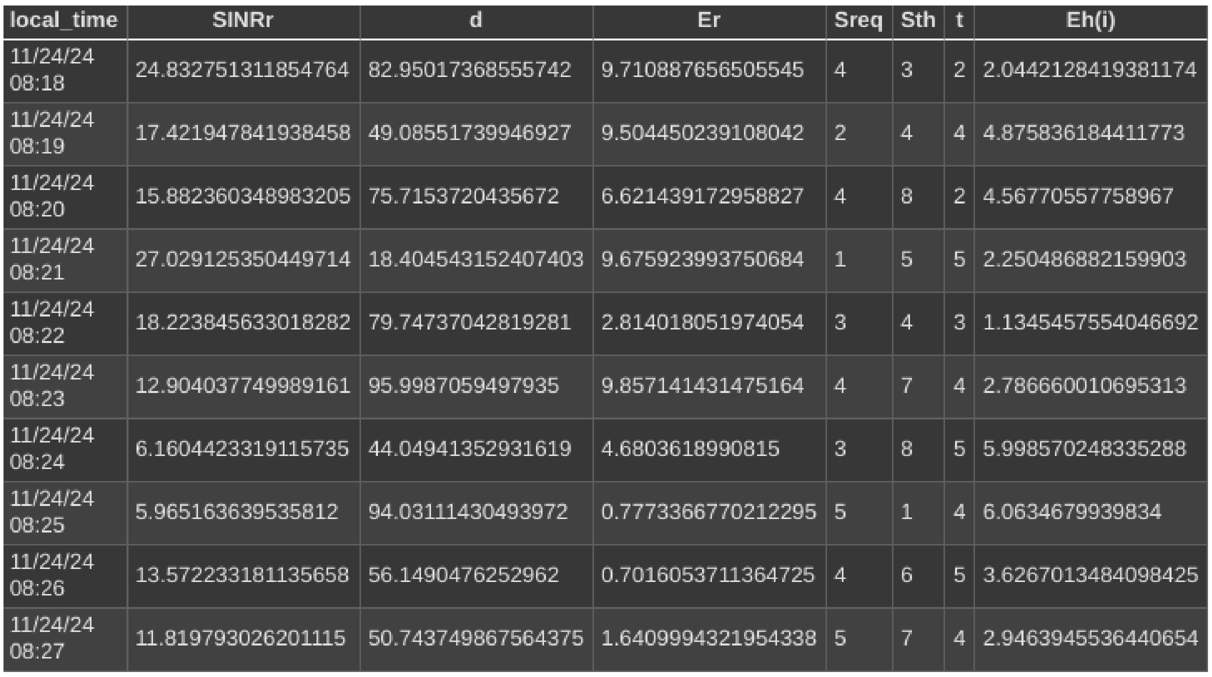
This dataset provides a comprehensive view of the UE's energy conditions, link quality, and security requirements, enabling the algorithm to balance energy efficiency and security by selecting an appropriate security suite.

- **local_time:** The timestamp of the data entry, representing when the data was recorded.
- **$SINR_r$:** The signal-to-noise ratio at the receiver, indicating the quality of the wireless link. A higher value suggests a stronger and more reliable signal, while a lower value indicates weaker signal quality.
- **d:** The distance between the user equipment (UE) and the base station, which affects the transmission power required and potentially the quality of the connection.
- **E_r:** The residual energy available at the UE at the start of the interval, reflecting the remaining battery level.
- **S_{req}:** The security requirement level for the UE in this interval. Higher values represent stricter security requirements, which consume more energy for encryption, authentication, and other security operations.
- **S_{th}:** The maximum allowable security threshold for the UE, determined by system or application constraints. A higher value permits more energy-intensive and secure

configurations, while a lower value restricts the energy consumption for security purposes.

- **t:** The duration of the time interval for which the algorithm evaluates security configurations.
- **E_h(i):** The predicted energy that the UE is expected to harvest during this interval, as estimated by the UKF algorithm.

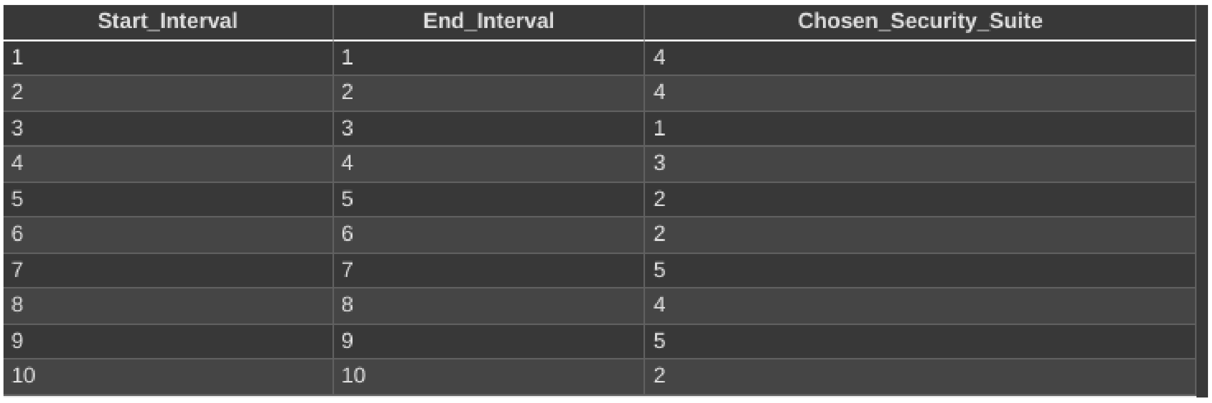
`ukf_energy_data.csv`



local_time	SINRr	d	Er	Sreq	Sth	t	Eh(i)
11/24/24 08:18	24.832751311854764	82.95017368555742	9.710887656505545	4	3	2	2.0442128419381174
11/24/24 08:19	17.421947841938458	49.08551739946927	9.504450239108042	2	4	4	4.875836184411773
11/24/24 08:20	15.882360348983205	75.7153720435672	6.621439172958827	4	8	2	4.56770557758967
11/24/24 08:21	27.029125350449714	18.404543152407403	9.675923993750684	1	5	5	2.250486882159903
11/24/24 08:22	18.223845633018282	79.74737042819281	2.814018051974054	3	4	3	1.1345457554046692
11/24/24 08:23	12.904037749989161	95.9987059497935	9.857141431475164	4	7	4	2.786660010695313
11/24/24 08:24	6.1604423319115735	44.04941352931619	4.6803618990815	3	8	5	5.998570248335288
11/24/24 08:25	5.965163639535812	94.03111430493972	0.7773366770212295	5	1	4	6.0634679939834
11/24/24 08:26	13.572233181135658	56.1490476252962	0.7016053711364725	4	6	5	3.6267013484098425
11/24/24 08:27	11.819793026201115	50.743749867564375	1.6409994321954338	5	7	4	2.9463945536440654

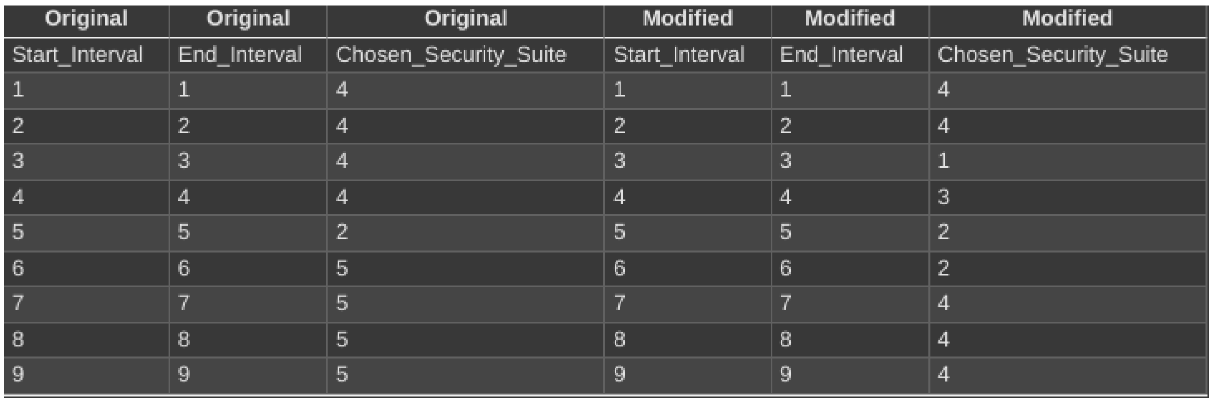
The following csv file captures the security suites assigned by the modified algorithm based on the energy availability and the set security threshold.

`chosen_security_suites.csv`



Start_Interval	End_Interval	Chosen_Security_Suite
1	1	4
2	2	4
3	3	1
4	4	3
5	5	2
6	6	2
7	7	5
8	8	4
9	9	5
10	10	2

`algorithm_comparison.csv`



Original	Original	Original	Modified	Modified	Modified
Start_Interval	End_Interval	Chosen_Security_Suite	Start_Interval	End_Interval	Chosen_Security_Suite
1	1	4	1	1	4
2	2	4	2	2	4
3	3	4	3	3	1
4	4	4	4	4	3
5	5	2	5	5	2
6	6	5	6	6	2
7	7	5	7	7	4
8	8	5	8	8	4
9	9	5	9	9	4

Future Works

Inclusion of Historical Data

Leveraging historical data about energy consumption patterns and past security requirements can help refine the calculation of the security threshold (S_{th}). This data could provide insights into trends and behaviours, enabling the system to anticipate future conditions and set thresholds more accurately.

Integration with Machine Learning Models

Using ML models can help predict future security requirements (S_{req}) and harvested energy ($E_h(i)$) based on environmental factors, user behaviour, and device usage patterns. This could make the algorithm more adaptive and efficient in varying scenarios.