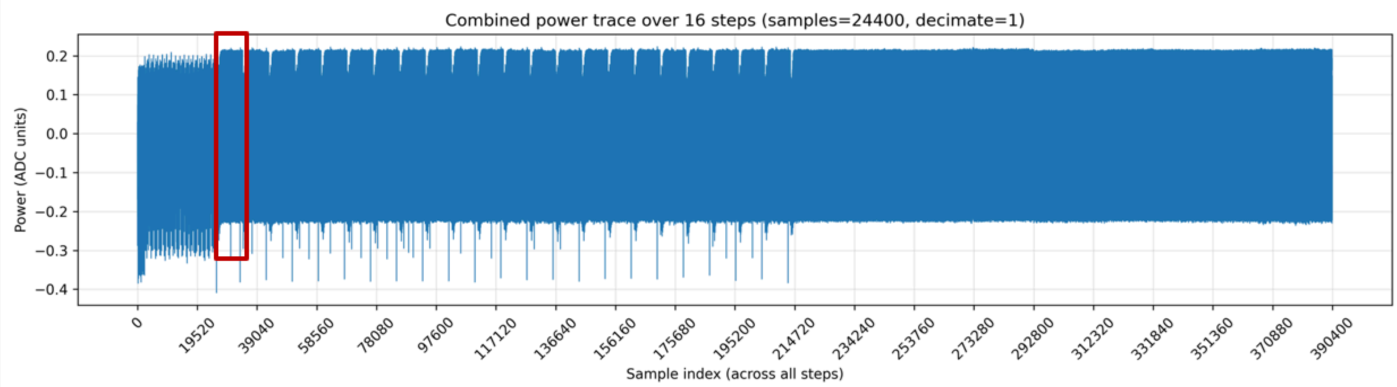
Task1

OVERVIEW

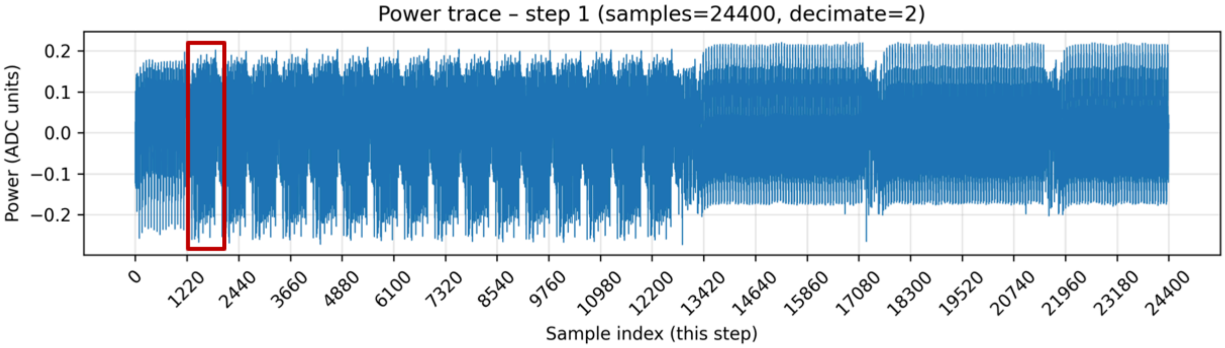
We know that on this board we have a buffer of 24,400 samples for measurement. To capture a longer execution, we can either lower the time resolution by increasing the decimation value, or keep a low decimation but repeat the measurement several times, each time increasing the offset to start recording from where the previous measurement ended. Then we concatenate all measurements to obtain a complete “helicopter view.”

You can check the `trace_steps_combined.png` file, which was created using this multi-step helicopter view method. In our Python code, we set the number of steps to 8 and the decimation to 2, and obtained the following result.

As shown in the figure, the red border highlights a repeating pattern that occurs 22 times. Since this pattern appears at the end of the trace, it cannot correspond to DES operations. Therefore, we interpret it as the UART transmission pattern.

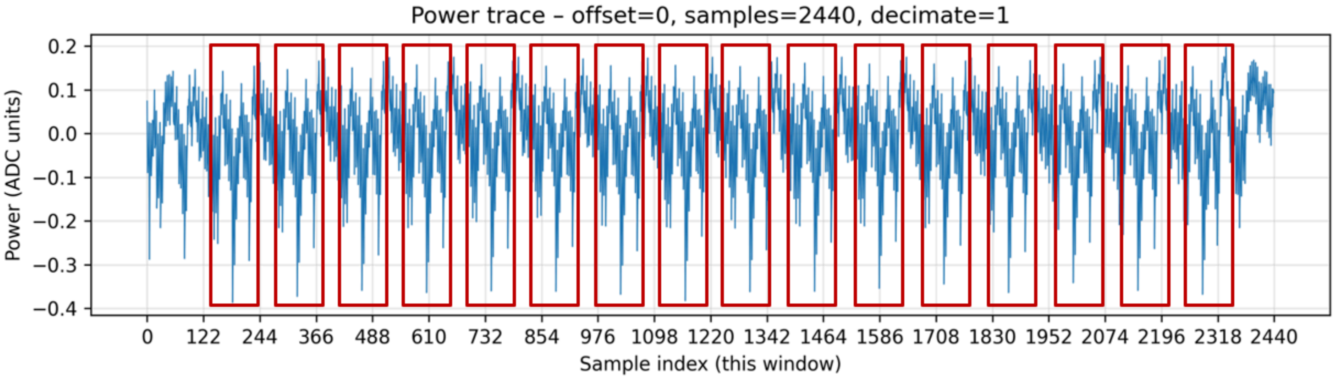


To identify the DES key generation and Feistel patterns, we zoomed in on the first part of the output. In the next figure, the red border highlights a pattern repeated 16 times. This likely corresponds to the Feistel steps in DES. Before this part of the measurement, we should find another 16 repeating patterns representing the key generation phase which is shown in the last figure.



In the next zoomed view, shown in the following figure, we can see another 16 repeated patterns that are shorter than the main encryption part. These shorter patterns correspond to the key generation of DES. This last image was captured using the `capture_window.py` script, which focuses on the early portion of the measurement.

Using this approach, we were able to identify all the patterns needed for key leakage analysis.


Power trace – offset=0, samples=2440, decimate=1
Power (ADC units)
Sample index (this window)