Defensive Phase:

Our solution to the challenge was to use a system of UID's to validate SED's in flight. The UIDs were randomly generated by the SSS and given to the SEDs during registration. Our hope was to prevent unauthorized devices from communicating with the properly provisioned SEDs since they would have a negligible chance of guessing a valid UUID. We also added randomly generated bytes to every message so that the attackers would not be able to gain information by repeating certain events that produce the same messages. Lastly, we also implemented a checksum to maintain the integrity of messages. The checksum was created using SHA-256, and was created after data of the message was encrypted. Our packets looked as follows:

| Header | Random | UUID | Data | Padding | Checksum |

The random, UUID, and checksum were always 6, 16, and 32 bytes respectively. The length of the data varied, so the padding was calculated for each message so that it was the correct length for AES encryption. This design did not work. We can assume that there was a flaw that allowed for communication between the properly provisioned SEDs and the spoofed one that allowed for attackers to successfully exploit our SCEWL network. We can build upon this design for the future by adding a sequence number to strengthen our defenses against replay attacks. Additionally, we can create a handshake with shared session keys so that our SEDs have another layer of encryption to protect our data.