Part 1: Search and Rescue

Monitor Mode

- Research and describe what monitor mode is: *Monitor mode is a special mode for wifi network interfaces that allows the adapter to capture all wireless traffic on a specific channel. Not just the packets addressed to it*
- In your write-up, describe why setting the channel is important: *Setting the channel is important because wifi operates across dofferent frequency channels (1-11 for 2.4GHz, and 36-165 in 5GHz). If two devices are communicating over two different channels, they are not gonna be able to hear each other even if they are close in proximity.*
- Provide terminal output from your script showing it successfully set monitor mode:

```
sudo ./set_monitor_mode.sh wlan0 11
[+] Setting wlan0 to monitor mode on channel 11...
[+] Verifying monitor mode...
        type monitor
        channel 11 (2462 MHz), width: 20 MHz (no HT), center1: 2462 MHz
[ ] wlan0 is now in monitor mode.
```

Beacon Design

- Describe what information your beacon contains that would help rescuers: *Each beacon frame carries as its payload a vendor-specific information element that contains the OUI (that identifies a frame as a frame coming from a survivor), and a uuid to identify each survivor uniquely. It also contains a RadioTap layer from which we can probe the strength of the signal and calculate the regression slope to find out whether we are getting closer or further*
- Describe when/how often you will transmit these beacons: *These becons are transmitted every one second which are not too fast that they would exhaust the battery of the transmitter, but not too slow that the rescuer wouldn't get timely feedback. The rate is also configurable and could be dynamically controlled to allow for a slower transmission when the survivor battery goes low*
- Discuss the idea of using RSSI as a proxy for distance and give examples of where it could go wrong: _Using RSSI to tell if you are getting close or further from a transmitter is generally reliable in short time windows if you move slowly and take several readings. However, due to multipath or reflections that could happen in buildings or rubble, consturctive (RSSI spikes) or destructive (RSSI drops) interferences that are unrelated to true distance could happen.

- Is RSSI a good proxy for distance in this rescue use case?: *It's an imperfect proxy for distance; It's good for directional guidance (following the gradient) but it's poor for absolute distance estimation. Since the system's goal is to locate a nearby person and not to compute a precise coordinate, it generally serves its function in this case. Especially when combined with human judgement.*
- VIDEO

Part 2: Secret Key Exchange

Calculate a key on each device

- Discuss reasonable values for z (the number of standard deviations).

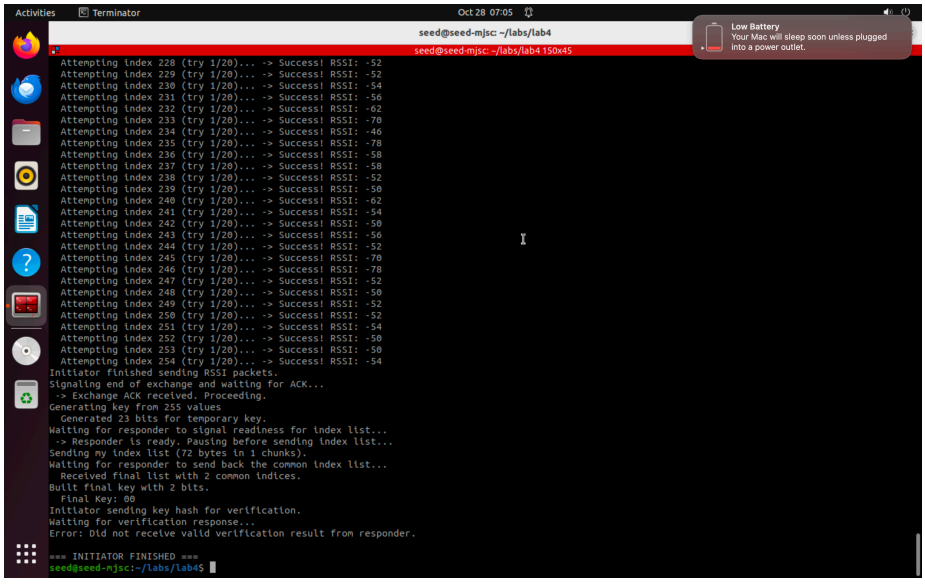
At first we started with relatively low (.75) values for z because we didn't want too many bits to be dropped, but we ended up having to go all the way up to 1.8 for the sake of consistency. The downside is of course much shorter keys.

- Describe how bits should be included in a cryptographic key that will serve as a basis for long-term secure communication.

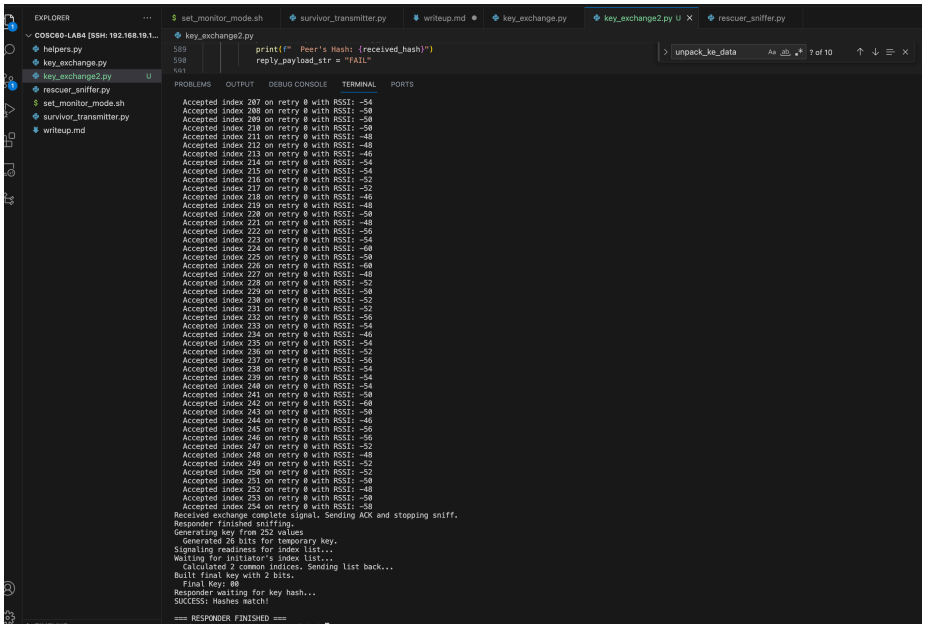
The bits from this lab aren't enough for a decent cryptographic key, but perhaps they could be used to seed key generation for both parties or to temporarily encrypt communication in order to store better long-term keys.

- Show evidence both devices independently computed the same key.

Initiator .



Responder .


EXPLORER
COSC60-LAB4 [SSH: 192.168.19.1...
helpers.py
key_exchange.py
key_exchange2.py
rescuer_sniffer.py
set_monitor_mode.sh
survivor_transmitter.py
writeup.md
set_monitor_mode.sh
survivor_transmitter.py
writeup.md
key_exchange.py
key_exchange2.py
rescuer_sniffer.py
key_exchange2.py
print(f" Peer's Hash: {received_hash}")
reply_payload_str = "FAIL"
unpack_ke_data
? of 10
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Accepted index 207 on retry 0 with RSSI: -54
Accepted index 208 on retry 0 with RSSI: -50
Accepted index 209 on retry 0 with RSSI: -50
Accepted index 210 on retry 0 with RSSI: -50
Accepted index 211 on retry 0 with RSSI: -48
Accepted index 212 on retry 0 with RSSI: -48
Accepted index 213 on retry 0 with RSSI: -46
Accepted index 214 on retry 0 with RSSI: -54
Accepted index 215 on retry 0 with RSSI: -54
Accepted index 216 on retry 0 with RSSI: -52
Accepted index 217 on retry 0 with RSSI: -52
Accepted index 218 on retry 0 with RSSI: -46
Accepted index 219 on retry 0 with RSSI: -48
Accepted index 220 on retry 0 with RSSI: -50
Accepted index 221 on retry 0 with RSSI: -48
Accepted index 222 on retry 0 with RSSI: -56
Accepted index 223 on retry 0 with RSSI: -54
Accepted index 224 on retry 0 with RSSI: -60
Accepted index 225 on retry 0 with RSSI: -50
Accepted index 226 on retry 0 with RSSI: -60
Accepted index 227 on retry 0 with RSSI: -48
Accepted index 228 on retry 0 with RSSI: -52
Accepted index 229 on retry 0 with RSSI: -50
Accepted index 230 on retry 0 with RSSI: -52
Accepted index 231 on retry 0 with RSSI: -52
Accepted index 232 on retry 0 with RSSI: -56
Accepted index 233 on retry 0 with RSSI: -54
Accepted index 234 on retry 0 with RSSI: -46
Accepted index 235 on retry 0 with RSSI: -54
Accepted index 236 on retry 0 with RSSI: -52
Accepted index 237 on retry 0 with RSSI: -56
Accepted index 238 on retry 0 with RSSI: -54
Accepted index 239 on retry 0 with RSSI: -54
Accepted index 240 on retry 0 with RSSI: -54
Accepted index 241 on retry 0 with RSSI: -50
Accepted index 242 on retry 0 with RSSI: -60
Accepted index 243 on retry 0 with RSSI: -50
Accepted index 244 on retry 0 with RSSI: -46
Accepted index 245 on retry 0 with RSSI: -56
Accepted index 246 on retry 0 with RSSI: -56
Accepted index 247 on retry 0 with RSSI: -52
Accepted index 248 on retry 0 with RSSI: -48
Accepted index 249 on retry 0 with RSSI: -52
Accepted index 250 on retry 0 with RSSI: -52
Accepted index 251 on retry 0 with RSSI: -50
Accepted index 252 on retry 0 with RSSI: -48
Accepted index 253 on retry 0 with RSSI: -50
Accepted index 254 on retry 0 with RSSI: -58
Received exchange complete signal. Sending ACK and stopping sniff.
Responder finished sniffing.
Generating key from 252 values
Generated 26 bits for temporary key.
Signaling readiness for index list...
Waiting for initiator's index list...
Calculated 2 common indices. Sending list back...
Built final key with 2 bits.
Final Key: 00
Responder waiting for key hash...
SUCCESS: Hashes match!
=== RESPONDER FINISHED ===