Authentication of BLE Wake-Up Reciever using RF Fingerprinting

Srivatsan Sridhar

Mentor: Mohamed Abdelhamid

Background

- Need for low power IoT receivers
  - Battery-operated IoT devices
  - In-body medical sensors
- Low power Wake-Up Receiver
  - Bluetooth Low Energy (BLE)
  - Duty-cycled operation
  - Wake up on detecting the wake-up pattern

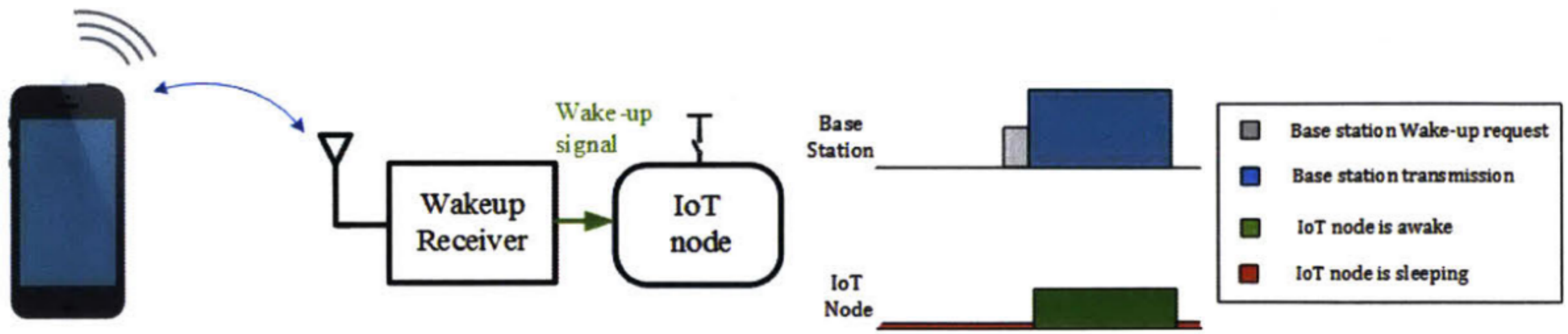


Figure 1-3: Wake-up receivers for IoT nodes

Motivation

- Security of the wake-up receiver
  - **Battery drainage attack** if wake-up pattern is leaked
  - Adversary can wake up the receiver by sending wake-up pattern

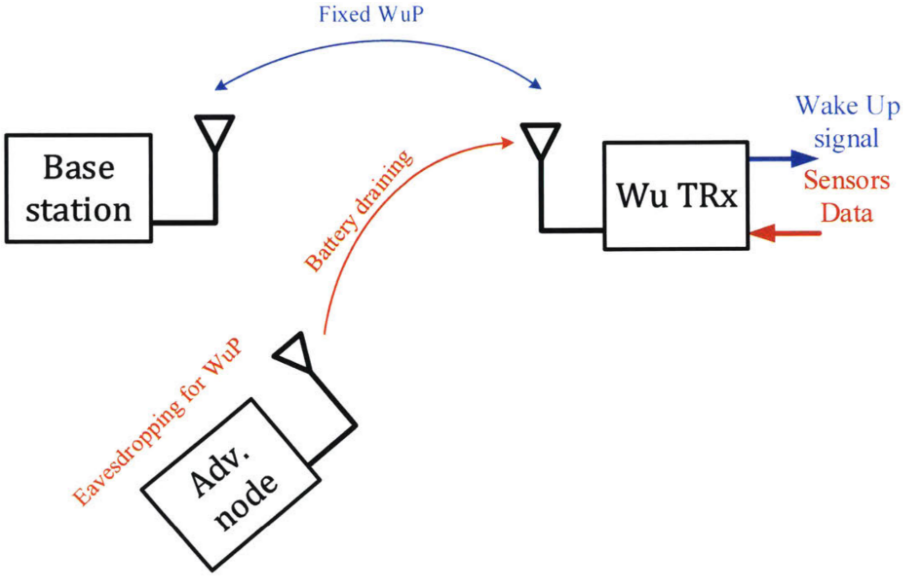


Figure 5-4: Battery drainage attacks in Fixed-pattern schemes

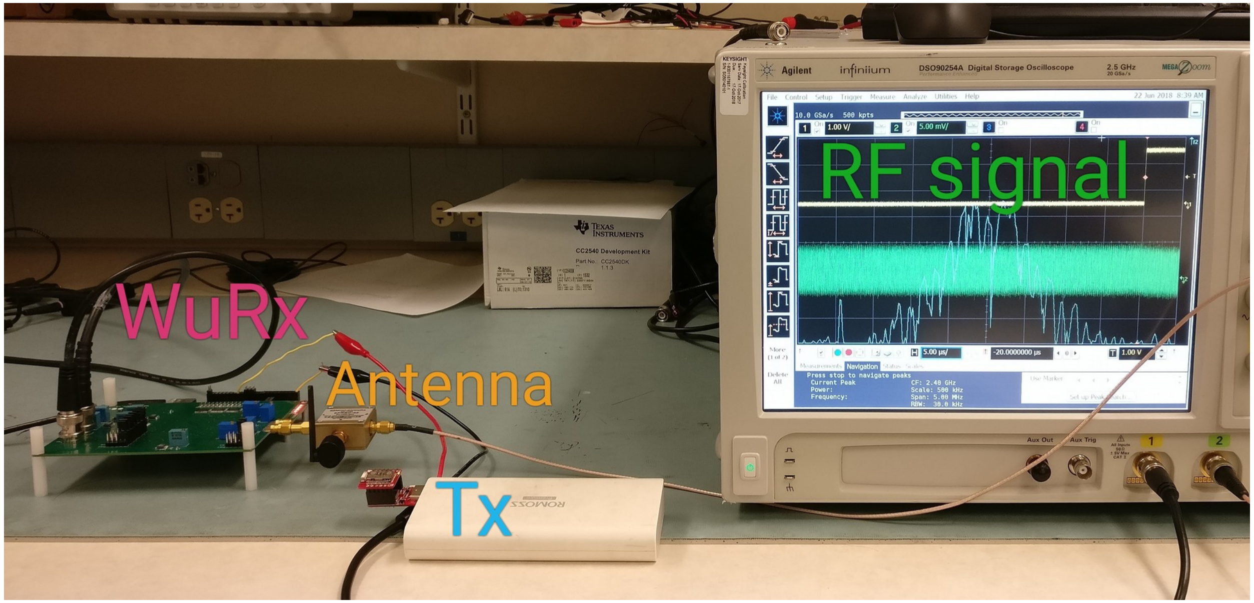
- Need for a **low power authentication system** for the wake-up receiver

Objectives

- Authentication of the wake-up receiver to prevent battery drainage attacks
- Accept wake-up patterns from only authorized transmitters
- **RF Fingerprinting:** Use unique features in the RF signal to identify the transmitter
- **Neural network for classification** of transmitter from features of the signal
- Acheive a reasonable identification accuracy while having a **low power implementation**

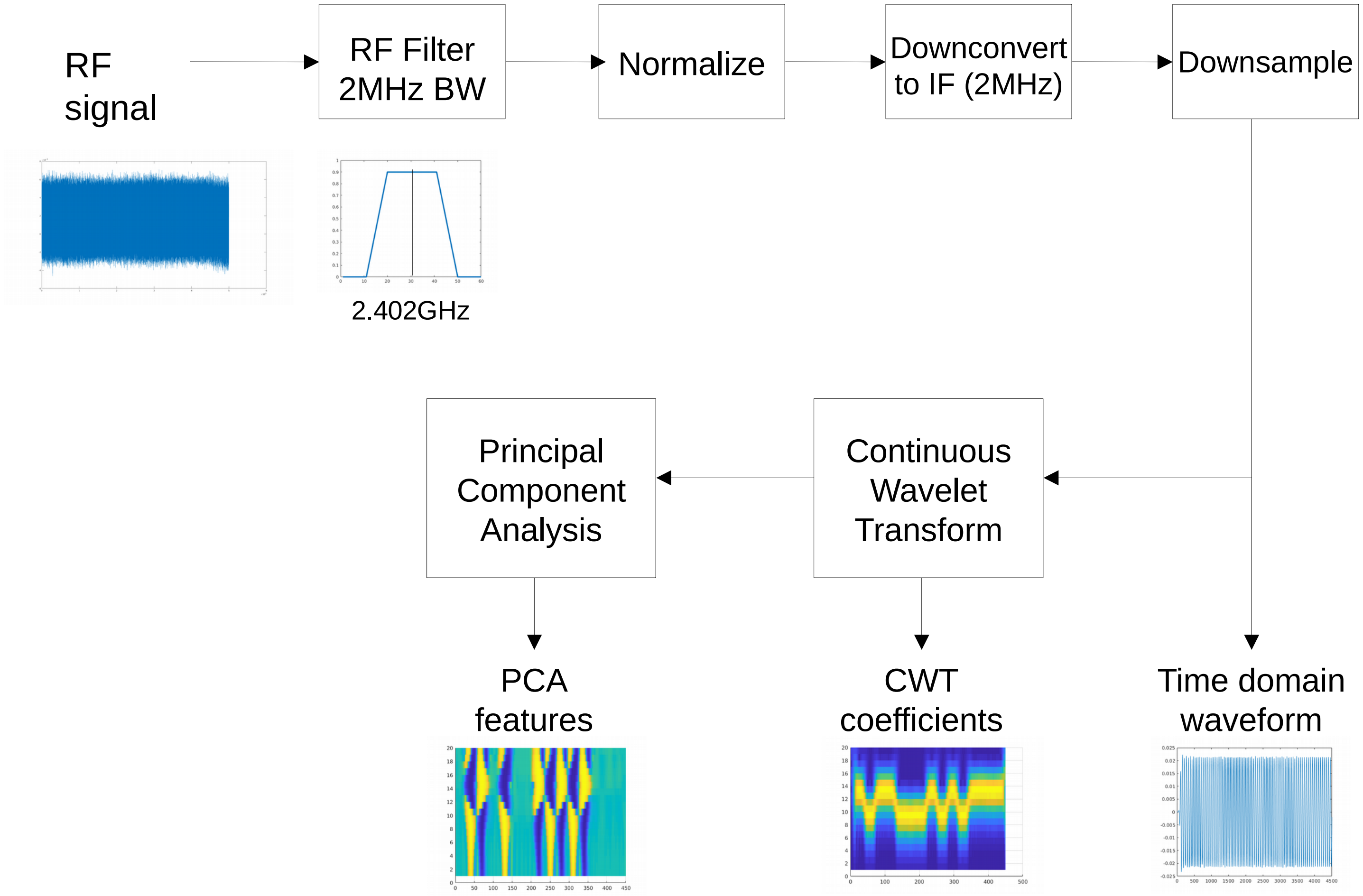
Experimental Setup

- Experimental Setup
  - Sampling Rate: 10 Gsamples/second
  - Collect 2000 waveforms each for **10 BLE transmitters**
  - **Fixed packet** (e.g. wake-up pattern) sent by all transmitters
  - **RF Signals** collected directly from the receiver antenna
  - Waveforms triggered using wake-up signal from wake-up reciever
  - Packet length: 45μs starting from the wake-up signal
  - Transmitter moved around in a radius of 30cm around the receiver



Signal Pre-processing

- Signal Pre-processing:



Signal Pre-processing

1) RF Filter
- IIR bandpass filter from 2.401 to 2.403 GHz

2) Down Conversion
- Multiply the signal by cosine of 2.4 GHz frequency

3) Down Sampling
- Downsample the signal by 100 times

4) Normalization
- Each time domain signal is made a unit L2-norm vector
- This vector is multiplied by 32 before 8-bit quantization

Signal Pre-processing

5) Continuous Wavelet Transform

- Uses Morlet wavelet. Keep absolute value of CWT
- Select only frequency bins from 1 to 4MHz (20 bins)
- Downsample the CWT along time axis by 10 times
- Quantize to 8 bits

6) Shuffling

- Shuffle the CWTs to generate a randomized dataset

7) Normalization

- Compute mean and std deviation of the first 16000 (training) CWTs
- Store mean and 4*std deviation, quantized to 8 bits
- From all CWTs, subtract mean and divide by std deviation
- Remove any NaNs or Infs after division
- Quantize normalized CWT*0.5 to 8 bits

Signal Pre-processing

8) Principal Component Analysis

- PCA coefficient matrices are computed on the training CWTs
- PCA vectors obtained by multiplying CWTs by these coefficients
- Use the 128 vectors with maximum variance
- Quantize PCA vectors after multiplying by suitabe scales

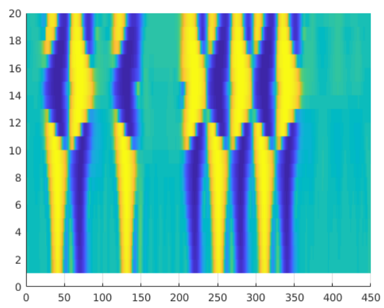
9) Quantization

- Every variable to be quantization is multiplied by a suitable scale to bring the values in [-1,1)
- Any values <-1 or >=1 are truncated to these limits
- These values are quantized to 8 bits with 1 sign bit and 7 bits for the fractional part
- Quantization is carried out on the result of every calculation

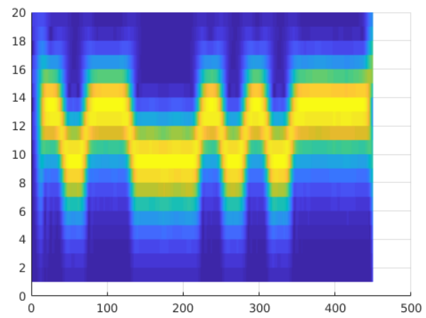
Neural Network Architecture

- **Input features**: Time domain signal or CWT coefficients or PCA features
  - CWT in frequency domain is useful because of FSK modulation
  - PCA helps to reduce the number of input features
  - Better accuracy obtained from CWT and PCA features

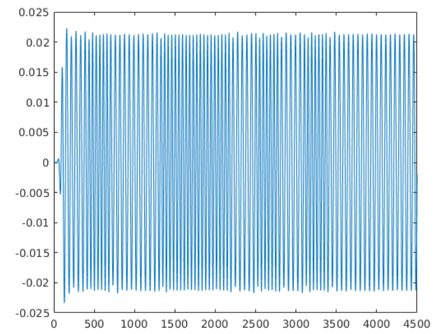
PCA features



CWT coefficients



Time domain waveform



- Two different network architectures:
  - **Convolutional neural network** – 2-D for CWT or 1-D for PCA
  - Single **fully connected layer** with dropout

- To have a **low power implementation**:
  - Minimize the **number of nodes** in the network
  - **Quantize** the inputs and weights to lesser number of bits

Neural Network Architecture

1) *Identification* – Identify which transmitter sent a signal
   - Input – CWT coefficients (80x20) as a 1-D vector
   - Single Hidden layer with 256 nodes, ReLU activation
   - Dropout rate of 0.3
   - Output layers with one node for each transmitter
   - Softmax cross entropy loss function
   - Weights and biases quantized to 8 bits after training

2) *Verification* – Verify if the signal was sent by a particular transmitter
   - Input – CWT coefficients (80x20) as a 1-D vector
   - Single Hidden layer with 256 nodes, ReLU activation
   - Dropout rate of 0.2
   - Output layers with one node, verify on positive value
   - Weighted sigmoid cross entropy loss (pos_weight = 0.5)
   - Weights and biases quantized to 8 bits after training

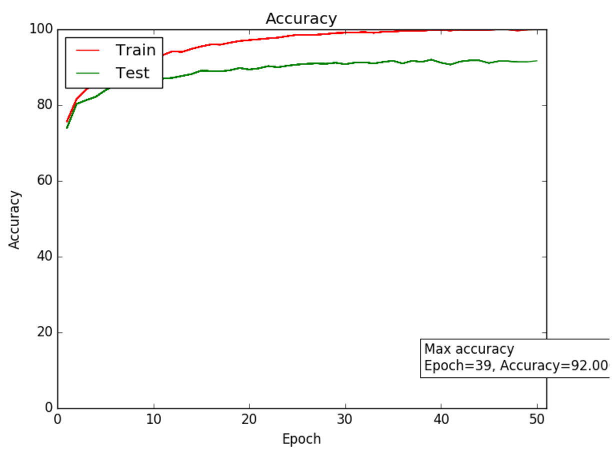
Discarded Approaches

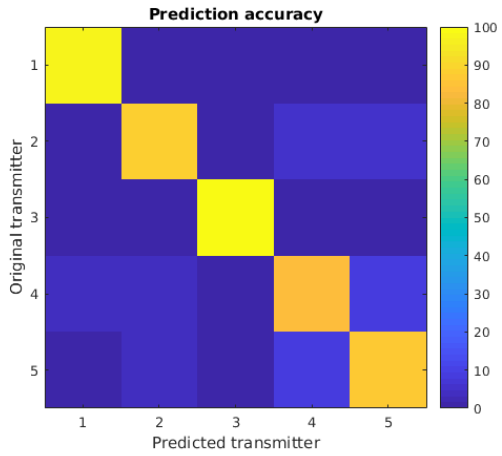
1) *RF signal from preamble*
   - Training data is the preamble of bluetooth packet
   - Variable length for each transmitter
   - Measured using an antenna very close to the transmitter
   - Classification accuracy
   - *Reason:* Use data from the wake-up pattern (packet) rather than preamble

2) *Principal Component Analysis*
   - Computed on the normalized CWT vectors
   - Helps to reduce size of input vectors to the neural network from 900 to 128
   - *Reason:* Accuracy similar to using just CWT but requires one additional matrix multiplication to compute the PCA

3) *Convolutional Neural Network*
   - Can help reduce number of weights that have to be stored
   - Ignores time and frequency shifts
   - *Reason:* Poorer accuracy than a single-layered neural network

Initital Achievements

- **Accuracy**
- Maximum classification accuracy of 92% using 5 transmitters
- Using one fully connected layer on CWT coefficients
- 



- **Minimizing the network**
- Achieved 90% accuracy with:
  - 128 input features obtained from PCA
  - Single fully connected layer of 128 nodes
  - 16-bit quantization of input features and network weights
- **Robustness**
- Position of transmitter upto 30cm around the receiver

Final Achievements

- **Identification with 10 transmitters**

Detected Transmitter

Actual Transmitter

	1	2	3	4	5	6	7	8	9	10
1	94.4	0	0	1.7	3.6	0	0	0.2	0	0
2	0.7	88.1	0	5.3	5.5	0	0	0	0	0.2
3	0	0	100	0	0	0	0	0	0	0
4	3.8	6.9	0	74.4	14.8	0	0	0	0	0
5	4.7	6.5	0	10.5	78.3	0	0	0	0	0
6	0	0	0	0	0	95.7	0	0.5	0.2	3.5
7	0	0	0	0	0	0.5	82.0	17.4	0	0
8	0	0	0	0	0	1.3	10.4	88.3	0	0
9	0	0.2	0	0.2	0	0	0	0	95.4	4.1
10	0	0	0	0	0	5.5	0	0	6	88.4

Confusion matrix for identification (% correctly detected for each transmitter)

- **Classification with 10 transmitters**

Transmitter no.	1	2	3	4	5	6	7	8	9	10
Precision	93.42	93.83	100	83.33	90.34	94.18	85	91.05	98.05	96.71
Recall	89.03	82.58	99.74	63.45	73.98	85	85.22	73.02	97.34	79.17

Precision = ‘true +ves’ / (‘true +ves’ + ‘false +ves’) in %
Recall = ‘true +ves’ / (‘true +ves’ + ‘false -ves’) in %

Conclusion and Future Work

- **Conclusion**
  - Reasonable identification accuracies
  - Suitable for the application of wake-up recievers
  - Accuracy varies with the transmitter
  - Small network and 8-bit values for low power consumption

- **Future Work**
  - Understand what are the features that are learned
  - Determine whether these features can be duplicated
  - Test on transmitters that were unknown during learning