POWER FINGERPRINTING IN UNAUTHORIZED SOFTWARE EXECUTION DETECTION FOR SDR REGULATORY COMPLIANCE

Carlos R. Aguayo González Jeffrey H. Reed

Wireless @ Virginia Tech
Bradley Department Electrical and Computer Engineering
Virginia Polytechnic Institute and State University

Wireless Innovation Conference
Nov. 30 - Dec. 3, 2010

Motivation for Detecting Unauthorized Software Execution

- Need for integrity assessment in SDR
  - An exploit can expose sensitive information
  - Disrupt critical communications infrastructure (due to access to wide spectral bands)
- Current approaches include:
  - Static techniques to identify vulnerability at compile time
  - Dynamic techniques based on formal methods or on software constructs to monitor execution behavior
  - Hardware-based approaches focused on tamper resistance and cryptography
    - Trusted computer module
    - Trusted Execution Technology

Limitations of Current Approaches

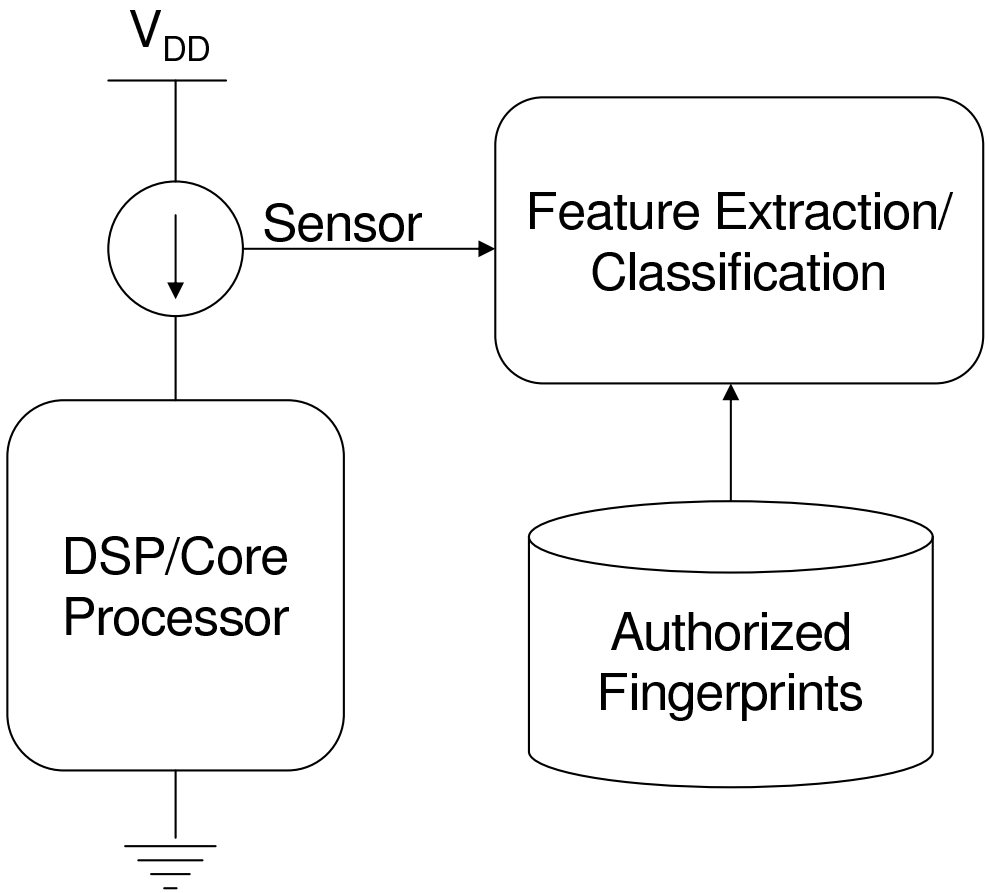
- These techniques only authenticate the loading of software, not its execution,
  - Devices are vulnerable to fault induction by unexpected environmental conditions and malicious insiders

- **Proposed Solution: Power Fingerprinting**

Power Fingerprinting

- In a processor dynamic power consumption depends on
  - Instructions being executed
  - Parameters
  - Addresses
  - Inter-instruction transitions
- A specific software routine will have a specific signature, or fingerprint
- Power fingerprinting can be used to monitor the actual execution of software
  - Extra layer of protection to complement existing approaches
  - Any unauthorized execution would disrupt power profiles
  - Very difficult to evade using traditional methods such as obfuscation and polymorphism

Proposed Approach



Power Fingerprinting Limitations

- Only provides integrity assessment and intrusion detection
  - it does not help with correctness or any other kind of validation.
- Requires a sensor to be physically connected to the board hosting the processor
- Discriminatory features are statistically in nature
  - Requires the observation of several execution cycles in order to provide a reliable assessment

Contributions for this Paper

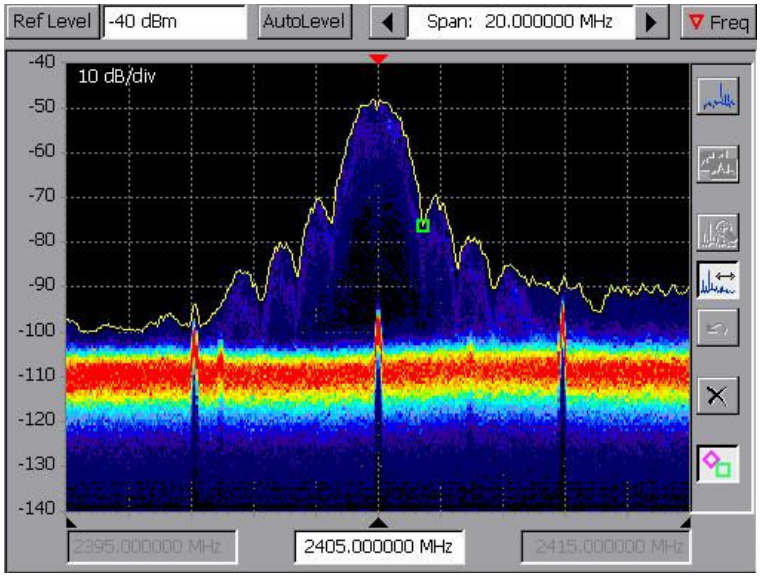
- Extend previous work on PF. Emphasis on regulatory applications
- Demonstrate ability to detect execution deviations that affects spectral emissions
- Detect tampering in configuration routines in a basic commercial radio platform

Test Platform Description

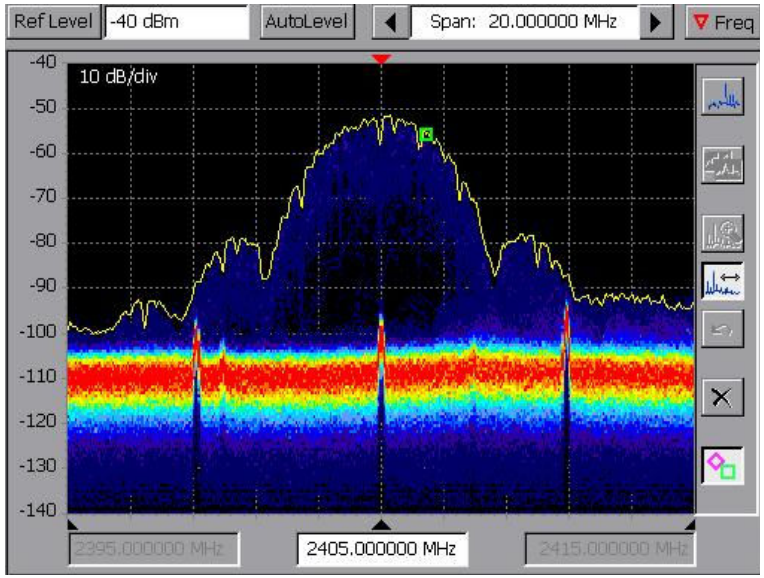
- PICDEM Z Demonstration Kit from Microchip Technology
  - Evaluation and development platform for IEEE 802.15.4
    - PHY and MAC layers for low-rate WPANs
    - In the 2450 MHz band, it supports data rates up to 250 kb/s
    - Basis for the ZigBee and MiWi
- The kit includes:
  - Motherboard with a PIC18LF4620 8-bit microcontroller
  - MRF24J40MA daughterboard with RF transceiver and antenna
  - Three different software stack implementations for Zigbee, MiWi, and MiWi P2P.
- We only use the MiWi P2P protocol (a variation of IEEE 802.15.4).
  - Uses only a portion of the IEEE 802.15.4 specification
  - No routing mechanism
  - Coverage is defined by the radio range

Operating Modes

Normal Mode (Default)



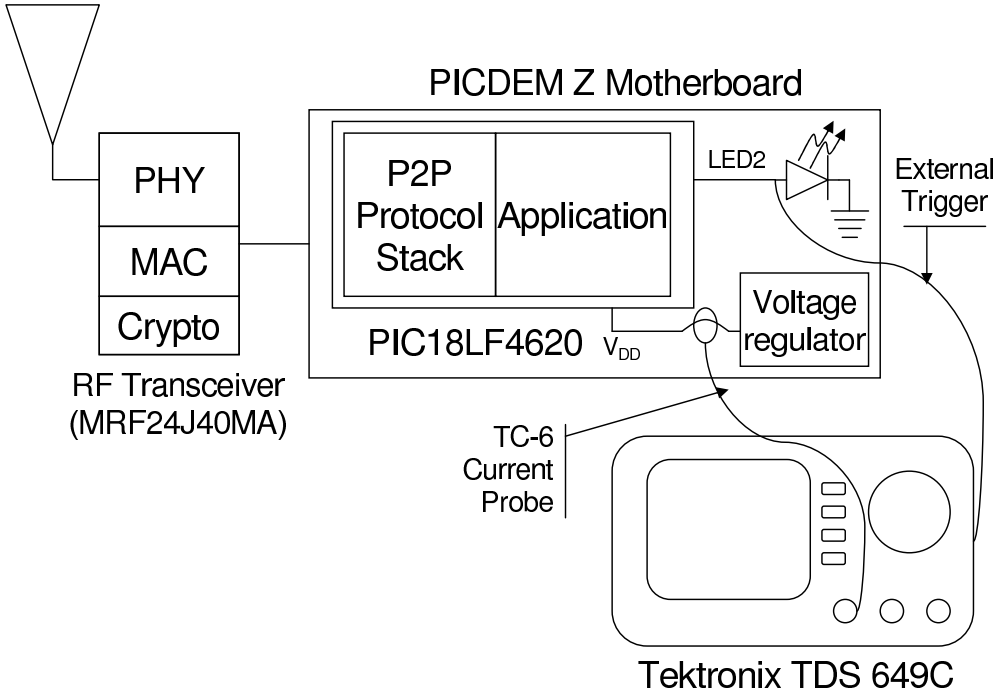
Turbo Mode



The “attacker” (a savvy user or an insider), could reconfigure the system looking to benefit from higher throughput

- Affects spectral emissions, invalidates the regulatory certification granted, and causes potential interference to other users

Measurement Setup



- Oscilloscope set to 500 MS/s and 10 mVΩ.
- $L = 30,000$ samples are collected every time.
- Six decoupling capacitors were removed (cumulative 6 μF).
- Current probe attached right after the voltage regulator (using a jumper)

Profiling Code and Test Scenarios

MiWi P2P Profiling Code

```
BoardInit();
ConsoleInit();
P2PInit();
SetChannel(myChannel);
EnableNewConnection();
CreateNewConnection();
FlushTx();
//write payload data
UnicastConnection(0,
                  FALSE,
                  TRUE);
//delay
_asm
    RESET   //Software Reset
_endasm
```

MiWi P2P Configuration Code

```
    .
    .
    .
TMR0H = 0x00; //Reset Timer 0
TMR0L = 0x00;
LED_2 = 1;     //Trigger
LED_2 = 0;

#ifdef TURBO_MODE
PHYSetShortRAMAddr(WRITE_BBREG0, 0x01);
PHYSetShortRAMAddr(WRITE_BBREG3, 0x38);
PHYSetShortRAMAddr(WRITE_BBREG4, 0x5C);

PHYSetShortRAMAddr(WRITE_RFCTL,0x04);
PHYSetShortRAMAddr(WRITE_RFCTL,0x00);
#endif
```

Two scenarios:

- Scenario 1: Default configuration vs TURBO_MODE preprocessor definition
- Scenario 2: Explicit Normal and Turbo configurations

Trace Analysis

Traces captured during the ith execution of α (code) are represented by

$$r_{\alpha}^{(i)}[n]; \qquad n = 0, \ldots, L-1$$

To avoid low-freq interference, the first difference between traces is used.

$$d_{\alpha}^{(i)}[n] = r_{\alpha}^{(i)}[n] - r_{\alpha}^{(i)}[n-1]$$

Trace Analysis (Cont.)

N traces are averaged to form the target signature and reduce the effects of random noise in our measurements.

$$s_\alpha[n] = \frac{1}{N}\sum_{i=0}^{N-1} d_\alpha^{(i)}[n]\,; \qquad n = 0, \ldots, L-1$$

Break signature onto j partial sections, each section has a length $w = \lfloor L/j \rfloor$.

Trace Analysis (Cont.)

The cross correlation for different sample lags, $0 \leq k \leq w$, of section j of the traces is given by:

$$\rho_{s_\alpha d_\beta^{(i)}}(j,k) = \frac{1}{(w-1)\sigma_s \sigma_d} \sum_{n=(j-1)*w}^{j*w} s_\alpha[n] d_\beta^{(i)}[k+n] - w\bar{s}\bar{d}$$

where $\bar{s}$ and σ_s are the sample mean and standard deviation of the corresponding section in s_α, respectively, and $\bar{d}$ and σ_d are the sample mean and standard deviation of the corresponding section in $d_\beta^{(i)}$.

Trace Analysis (Cont.)

Keep only the maximum correlation values for different lags.

$$\hat{\rho}_{s_\alpha r_\beta^{(i)}}(j) = \max_k \left\{ \rho_{s_\alpha r_\beta^{(i)}}(j,k) \right\}$$

If $\beta = \alpha$, then $\hat{\rho}_{s_\alpha r_\beta^{(i)}}(j) = 1$ for every section j.

We are interested only on the minimum peak correlation value for that specific trace

$$x_\beta^{(i)} = \min_j \hat{\rho}_{s_\alpha r_\beta^{(i)}}(j)$$
$$X_\beta = x_\beta^{(i)}; \qquad \forall i$$

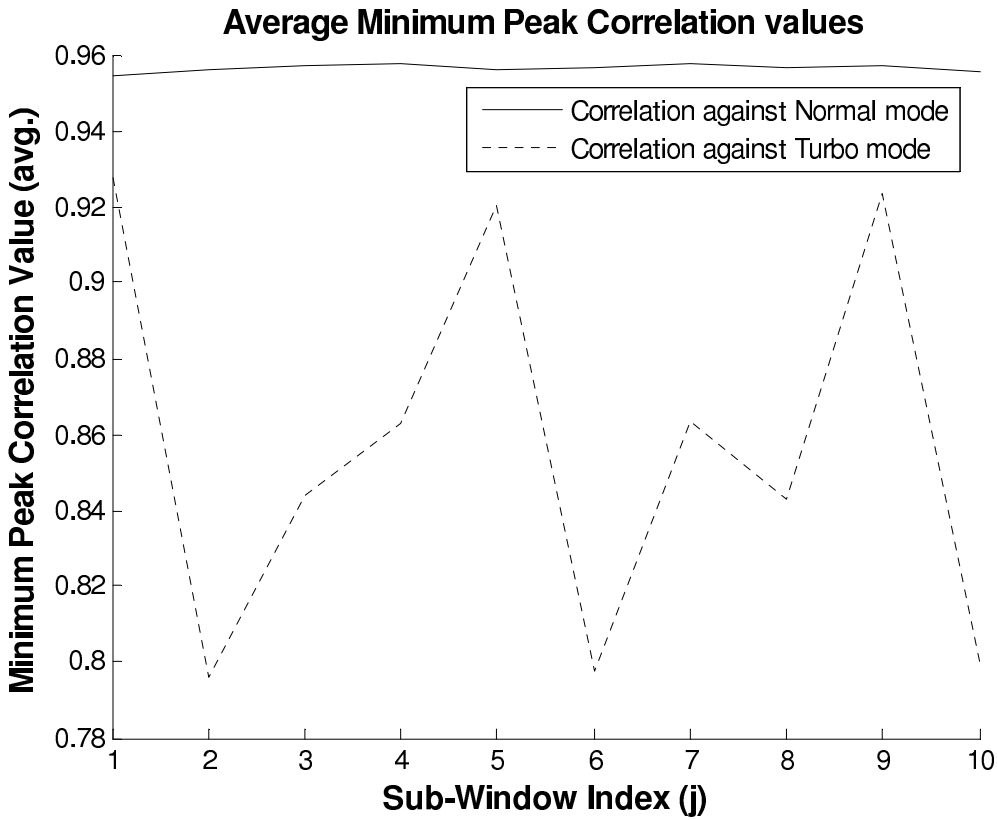
Trace Analysis Summary

After capturing traces:

- divide them into j subsections
- align each section against the corresponding section of the target signature
- calculate the correlation coefficient between them
- determine the maximum deviation from the signature
- make a decision whether the traces correspond to the execution of the target code

Correlation Results: Scenario 1

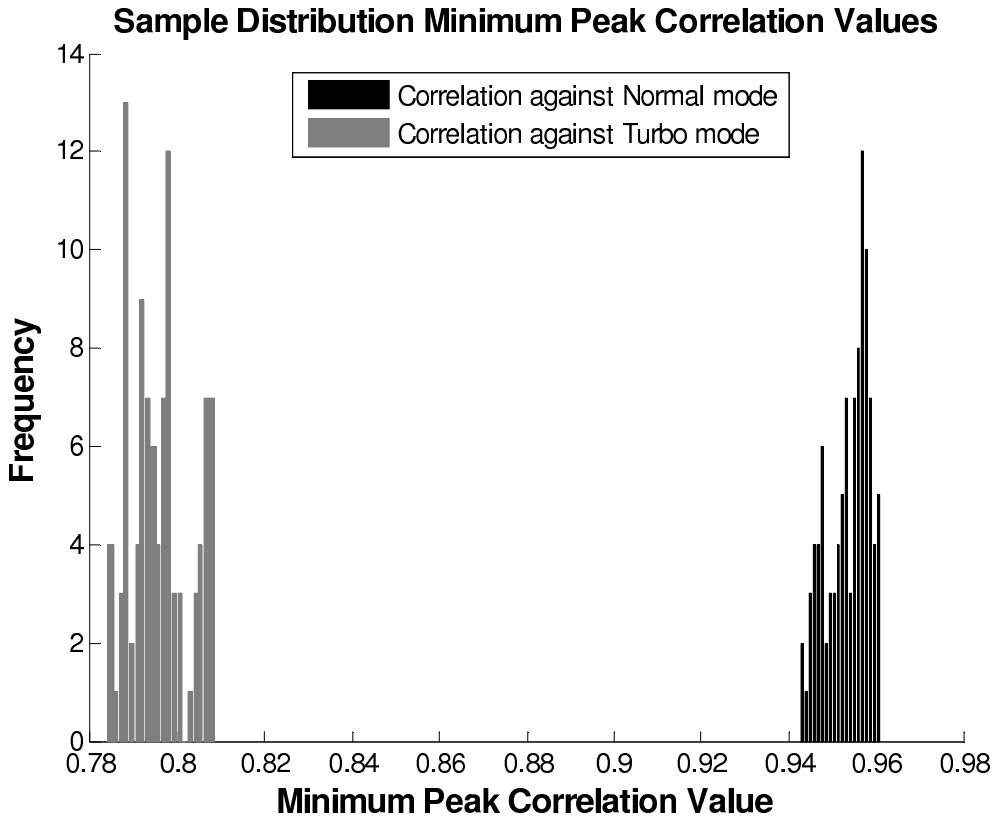
The average peak correlation values is given by $\bar{\rho}_{s_\alpha r_\beta}(j) = \frac{1}{N}\sum_{i=1}^{N} \hat{\rho}_{s_\alpha r_\beta^{(i)}}(j)$



Average minimum peak correlation value when original code uses default configuration

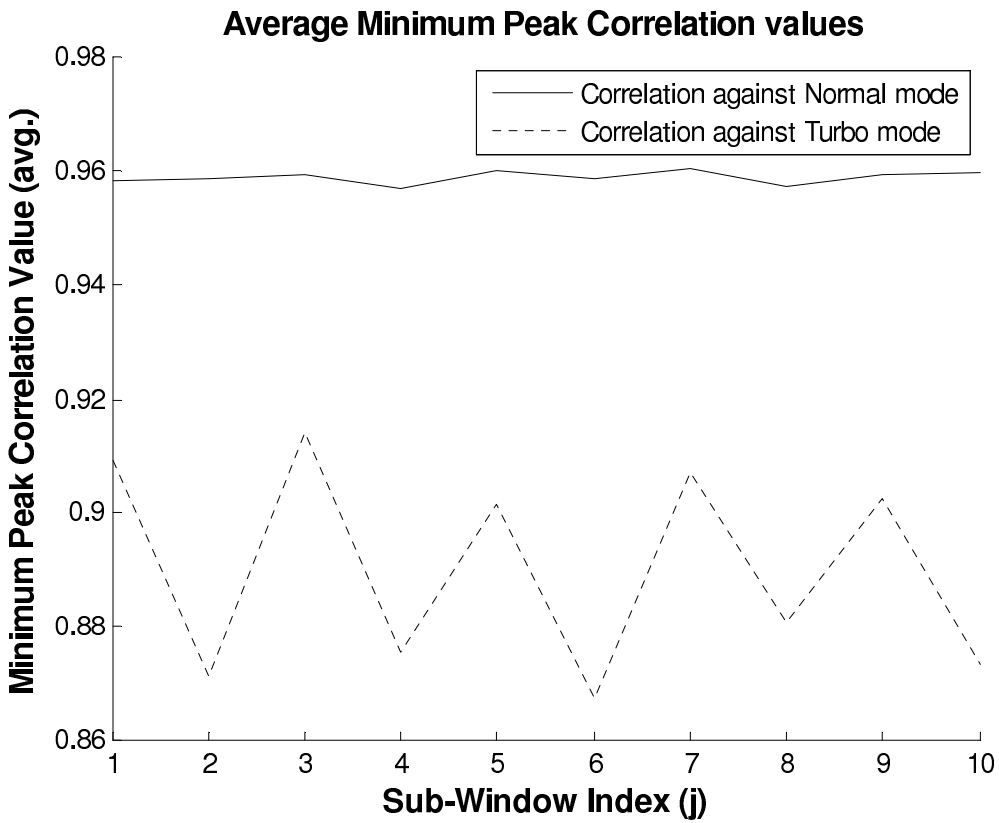
Sample Distribution Correlation Results: Scenario 1

The distribution of the minimum peak correlation values for all traces captured in both scenarios, X_α and X_β



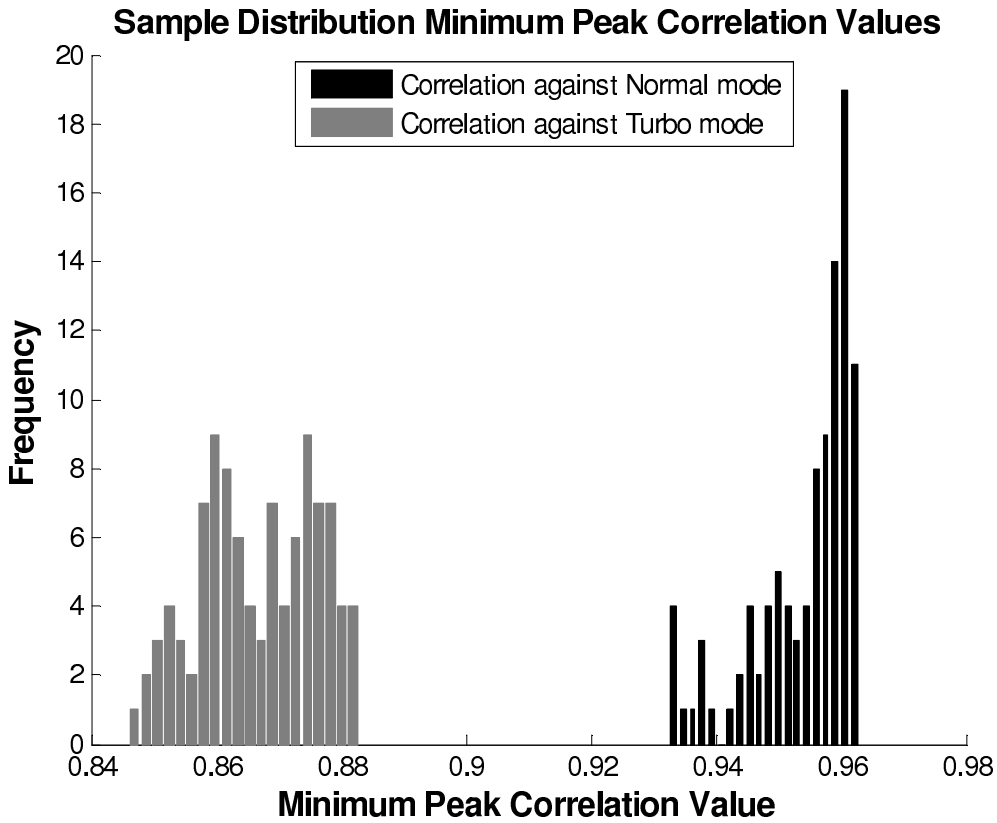
Minimum peak correlation values sample distribution when original code uses default configuration

Correlation Results: Scenario 2



Average minimum peak correlation value when original code uses explicit configuration

Sample Distribution Correlation Results: Scenario 1



Minimum peak correlation values sample distribution when original code uses explicit configuration

Conclusion and Future Work

- Power fingerprinting can be a useful tool in SDR regulatory enforcement (in feasible platforms)
- Demonstrated the ability of power fingerprinting to detect execution deviations that impact spectral emissions
- Lots of work still needs to be done

Thank you

The authors would like to thank Wireless @ Virginia Tech Affiliates for their support. This work was supported in part by the National Science Foundation under Grant CNS-0910531. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or Wireless @ Virginia Tech Affiliates.