POWER FINGERPRINTING IN UNAUTHORIZED SOFTWARE EXECUTION DETECTION FOR SDR REGULATORY COMPLIANCE

Carlos R. Aguayo Gonzalez (MPRG, Wireless@Virginia Tech, Blacksburg, VA, USA; caguayog@vt.edu); and Jeffrey H. Reed (MPRG, Wireless@Virginia Tech, Blacksburg, VA, USA; reedjh@vt.edu).

ABSTRACT

Software-defined radios (SDRs) improve flexibility and reduce development and maintenance costs in modern wireless communication systems. Unfortunately, they also present increased interference risks as a result of their ability to access wide spectral bands and their vulnerability to malicious software attacks. Hence, monitoring execution integrity is a critical element in regulatory policy for SDRs.

Power fingerprinting (PF) has been proposed as a technique to assess the integrity of SDRs and detect the execution of unauthorized software. PF obtains execution status information from the dynamic power consumption of the processor and extracts discriminatory features from it to determine whether the execution is authorized. In this paper, we provide further evidence of the ability of PF to detect unauthorized execution in SDR. Our results show that PF is able to detect an unauthorized change in the configuration of a PICDEM Z evaluation board which doubles the occupied transmission bandwidth.

1. INTRODUCTION

Software defined radio (SDR) delivers unseen flexibility in communications systems, reducing time to market, facilitating upgrades, and enabling exciting new technologies, such as cognitive radio. This flexibility, however, also brings difficult challenges in terms of spectral regulation and interference risk management. A malfunctioning radio (due to misconfiguration, malfunction, of a malicious attack), can cause interference in wider frequency bands and disrupt critical communication systems. Regulatory bodies, such as the US Federal Communications Commission (FCC), have recognized this challenge and require radio manufacturers, as part of the certification process, to demonstrate that no unauthorized software that affects spectral emissions can be executed in a deployed radio. The FCC, however, did not define how to accomplish this and it is left to the manufacturers to select a technique to maintain software integrity. Power fingerprinting (PF) has been proposed as a technique to monitor the integrity of SDRs and detect the execution of unauthorized software. In this paper we provide further evidence to support the application of power fingerprinting in SDR for regulatory applications.

Dynamic power consumption in a digital processor is due to transient currents and load capacitances charge and discharge that occur during bit transitions [1]. These transitions depend on the specific instructions being executed, parameters, and addresses. Therefore, the execution of a given software routine yields a specific power consumption pattern, or fingerprint. Any deviation from authorized execution, such as malicious intrusion or bypass of a critical module, will disrupt the fingerprints.

Using PF we can develop an external monitoring device which is constantly looking for deviations from the fingerprints of authorized code. This monitor can be used to enforce the exclusive execution of certified software in SDR. *During device certification, the power fingerprint of the certified software is stored independently and then used at runtime to verify that only the software used during certification is executed*. Regulatory entities can improve interference risk management using this approach and promptly stop execution when critical software that affects spectral emissions is modified after deployment.

In this paper, we extend previous work on PF. We provide further experimental evidence of the ability of PF to detect deviations from authorized code that affects spectral emissions. In our experiments, we successfully detect configuration changes that cause a violation in the spectral emissions of a basic commercial radio platform from Microchip.

2. PREVIOUS WORK

Run-time execution status monitoring is traditionally accomplished by inserting small software constructs and reporting tasks in the target system, as described in [2]. A similar approach is presented in [3] with the addition of an expressive specification method for defining time constraints. These traditional schemes, however, suffer from the observer's effect and depend on the same processor to perform the monitoring tasks.

In terms of software integrity, most of the literature is focused on the host domain (e.g. personal computers). Current approaches to address software integrity include: static techniques (identify vulnerability at compile time), dynamic techniques (based on formal methods or on software constructs to monitor execution behavior), and hardware-based approaches (tamper resistance and cryptography). Examples of hardware-based approaches include the Trusted Computing Module (TCM) and Intel's Trusted Execution Technology. The former performs load time attestation using hash functions and secure key management and storage. The latter builds from the TCM and provides hardware-based protected execution and memory spaces. Unfortunately, the collision resistance of some hash functions has already been compromised [4]. Furthermore, these techniques only authenticate the loading of software, not its execution, leaving devices vulnerable to fault induction by unexpected environmental conditions. Our proposed approach monitors the actual execution of software,

providing an extra layer of protection to complement existing approaches.

Using dynamic power consumption to obtain software execution status is the basis for power analysis side-channel attacks. In these attacks, dynamic power consumption is analyzed to obtain cryptographic keys and other secret information from digital devices [5]. Because dynamic power consumption is directly related to the computations being performed, attackers can use this information to break what could be considered mathematically strong algorithms. The same phenomena that allow attackers to "see" inside a cryptographic device, is used in power fingerprinting to assess the integrity of the code we expect to be executing in a processor. Previous results in [6], [7] showed the ability of power fingerprinting to detect execution deviations from a simple routine that toggled the contents of a register. In this paper we extend the previous results by demonstrating the ability of PF to detect execution variations that can potentially impact spectral emissions.

3. APPROACH DESCRIPTION

The general structure of the proposed PF monitor is depicted in Fig. 1. A sensor is placed between the power supply and the main processing hardware to measure the instantaneous current drain of the processor. The sensor should be placed as close to the core processor as possible to reduce interference caused by other components on the board. For our experiments, we use a current probe, but other approaches such as a current mirror can be used [8].

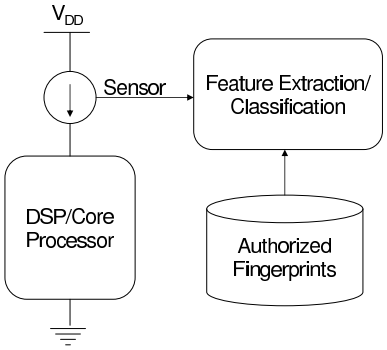

Fig. 1. Proposed system description

The most critical aspect in PF is the selection and extraction of discriminatory features from power traces. Correct integrity assessment depends on the discriminatory qualities of the features selected. Unfortunately, there is no effective procedure to identify optimal features. For this work we use simple time-domain correlation, but other approaches (e.g Euclidean distance, energy ratios, wavelet analysis) can be applied in different domains (e.g. frequency and cyclic-frequency).

Once the discriminatory features are extracted from the power traces, a suitable detector compares them with the expected signatures to determine whether the current execution is authorized or not. In our case a simple detector using the Neyman-Pearson criteria is designed, but different approaches, including neural networks and support vector machines, can be applied.

4. EXPERIMENT DESIGN

The goal for our experiments is to determine whether power fingerprinting is able to detect deviations from authorized software that affect spectral emissions. In order to accomplish this, we select a basic commercial radio platform which can be configured in two different operational modes: **Normal Mode**, which will be considered allowed, and **Turbo Mode**, which allows higher throughput but uses 2.5 times the bandwidth of the Normal Mode. The goal is to identify when the Turbo Mode is configured instead of the Normal Mode.

4.1. Hardware Platform

Our target platform is a PICDEM Z Demonstration Kit from Microchip Technology Inc. [9]. This kit is intended as an evaluation and development platform for IEEE 802.15.4 [10] application designers. This standard describes the physical (PHY) and media access control (MAC) layers for low-rate wireless personal area networks (WPANs) implemented using low-cost devices. In the 2450 MHz band, the standard defines a PHY layer using direct-sequence spread spectrum (DSSS), supporting an over-the-air data rate up to 250 kb/s. The 802.15.4 standard is the basis for the ZigBee and MiWi specifications, which define upper layers not covered in the standard to provide a complete networking solution. The kit includes a motherboard with a PIC18LF4620 8-bit microcontroller, a MRF24J40MA daughterboard with RF transceiver and antenna, and three different software stack implementations for Zigbee, MiWi, and MiWi P2P.

The MRF24J40MA in its **Normal Mode** implements the IEEE 802.15.4 specifications PHY layer. It employs a 16-ary quasi-orthogonal modulation technique and direct-sequence spread spectrum with a chip rate of 2000 Mchips/s. Four informations bits are used to select one of 16 nearly-orthogonal pseudo-noise sequences (PN) to be transmitted. The sequences are 32 bits long yielding a spreading gain of 8. The PN sequences for successive data symbols are concatenated, and the aggregate chip sequence is modulated onto the carrier using offset quadrature phase-shift keying (O-QPSK). Figure 2 shows a snapshot of the spectrum usage during a transmission using this normal (default) configuration.

The module, however, also implements a **Turbo Mode** which can deliver a maximum throughput of 625 kbps. This is an increase of 2.5 times the IEEE 802.15.4 specifications. The Turbo Mode is not compliant with the IEEE specifications and no information is given about the specific modifications that yield the Turbo Mode. After observing the occupied spectrum, however, it is clear the throughput improvements come as a result of wider spectrum usage, as can be seen in Figure 3. Notice how the spectrum occupancy is also roughly 2.5 time the occupancy in the IEEE 802.15.4 standard.

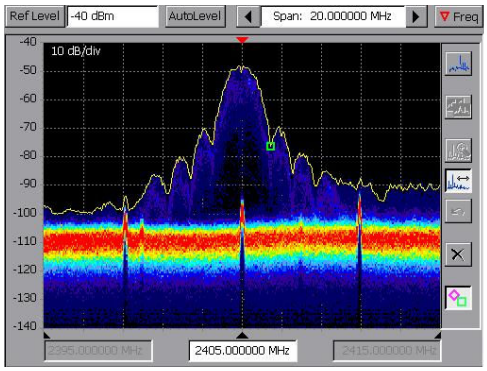

Fig. 2. Normal mode spectral occupancy

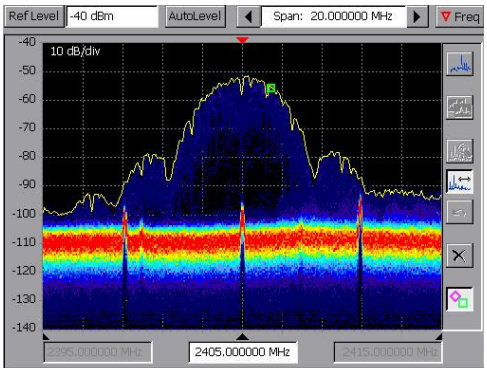

Fig. 3. Turbo mode spectral occupancy

4.2. Different Test Scenarios

The two possible configurations of the MRF24J40MA RF module are used in our experiment to test whether power fingerprinting can detect execution changes that impact spectral emissions. The Normal Mode, as specified in the IEEE 802.15.4 standard, is considered our authorized configuration. We consider the following scenario: A product using the MRF24J40MA configured by a PIC processor is approved for market release by a regulatory body under the Normal Mode. A savvy user, or an insider, (the "attacker") looks to benefit from higher throughput and reconfigures the system to use the Turbo Mode. This modification affects spectral emissions, invalidates the regulatory certification granted, and can potentially cause interference to other users. The actual attack model is irrelevant. So, we assume that the attacker has access to the source code and a way to reprogram the PIC processor configuring the RF module.

Once again, our goal is to use PF to detect an execution violation during the configuration of the RF module. It is assumed that a power fingerprint is extracted at certification time from a device configuring the Normal Mode. A power fingerprinting monitor will capture traces at run-time and determine whether the execution corresponds to the code use during certification or if tampering has occurred.

4.3. Profiling Software

We use a simple software structure to test the previous scenarios. The test code, shown in Listing 1, uses the set of APIs provided by Microchip as part of the implementation of the MiWi P2P Wireless Protocol stack.

In this routine, the board is initialized, a connection is established, the RF transceiver sends a packet using the UnicastConnection API, and a software reset is performed after a short delay.

Listing 1. MiWi P2P Profiling Code

```
1  BoardInit();
2  ConsoleInit();
3  P2PInit();
4  SetChannel(myChannel);
5  EnableNewConnection();
6  CreateNewConnection();
7  FlushTx();
8  //write payload data
9  UnicastConnection(0, //ID
10                    FALSE,//Cmd?
11                    TRUE);//crypto?
12 //delay
13 _asm
14     RESET  //Software Reset
15 _endasm
```

Within the P2PInit() implementation, there is a call to another function called MRF24J40Init() at the end of which the code shown in Listing 2 is included. The three lines writing to the different base-band registers (BBREG0, BBREG3, BBREG4) in the MRF24J40 configure it in the Turbo Mode. The last two lines, perform a reset of the RF state machine so the changes are implemented by the MRF24J40. The default configuration of the MRF24J40 upon reset is the Normal Mode.

Listing 2. MiWi P2P Configuration Code

```
1     .
2     .
3     .
4  TMR0H = 0x00; //Clear Timer0 high byte
5  TMR0L = 0x00; //Clear Timer0 high byte
6  LED_2 = 1;    //Falling edge in LED2
7  LED_2 = 0;    //to trigger oscilloscope
8
9  #ifdef TURBO_MODE
10     PHYSetShortRAMAddr(WRITE_BBREG0, 0x01);
11     PHYSetShortRAMAddr(WRITE_BBREG3, 0x38);
12     PHYSetShortRAMAddr(WRITE_BBREG4, 0x5C);
13
14     PHYSetShortRAMAddr(WRITE_RFCTL,0x04);
15     PHYSetShortRAMAddr(WRITE_RFCTL,0x00);
16 #endif
```

We test two different scenarios. In Scenario 1, we aim to discriminate between code compiled with and without the `TURBO_MODE` definition. The attacker, then, would just have to recompile using the `TURBO_MODE` directive to set up the Turbo Mode. The authorized version does not include the code within the `#ifdef` section.

Notice that in Scenario 1, extra instructions are inserted for configuring the Turbo Mode. This makes the job of the PF monitor easy, as the insertion changes instructions, fetches, and parameters, therefore yielding a much different fingerprint.

In Scenario 2, however, the Normal Mode configuration is explicit. The `#ifdef` directives are removed and the Normal

Mode is configured by setting the values of BBREG0, BBREG3, BBREG4 to 0x00, 0xD8, and 0x9C, respectively. The attacker still has to write the base-band registers with the values in Listing 2 to set the Turbo Mode. Scenario 2 makes power fingerprinting more challenging, as there are no instructions added, just a change in the parameters.

In order to make power trace capture of the configuration code more efficient, we execute a software reset, after a short delay, once the unicast transmission is completed. This does not affect the execution of the profiled code.

There are other important aspects of the profiling code that were included to facilitate trace captures. Right before executing the configuration code, we restart Timer 0 in the processor and turn on and off LED 2 in the board. Timer 0, a 16-bit timer, is used by the MiWi P2P software stack and is always running. By reinitializing it, we have a more deterministic power signature without affecting the operation of the system. LED 2 is used to trigger trace capture.

4.4. Measurement Setup

Trace collection is performed using a Tektronix TDS 649C real-time oscilloscope and a Tektronix CT-6 current probe. The probe is connected right after the voltage regulators on the mother board, which actually has embedded provisions for doing this kind of measurements. The oscilloscope is configured to 500 MS/s and 10 mVΩ. The trigger is configured to external source (driven by LED2), falling-edge, 40 mV level, and no pre-trigger samples are kept. A total of $L = 30,000$ samples are collected after every trigger event. The experiment setup is depicted in Fig 4. Two hundred traces are captured from the execution of each scenario and transfered to a host computer using GPIB for their posterior analysis in MATLAB.

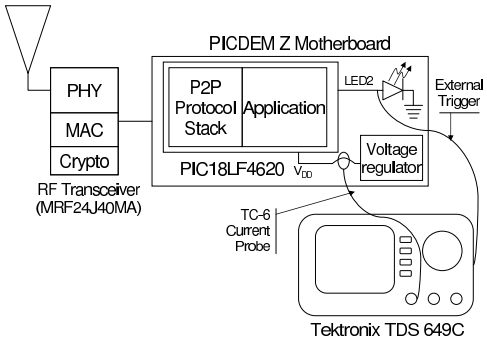

Fig. 4. Measurement Setup

The board itself was slightly modified to let power consumption features be captured. From the motherboard, a total of six decoupling capacitors were removed totaling a cumulative 6 μF. The function of these capacitors is to mitigate the stress placed on the power supplies by the strong current peaks caused by digital processors. It is important to note that removing decoupling capacitors would not be necessary if the current sensor is placed closer to the processor power pins.

4.5. Feature Extraction

Trace analysis to extract signatures and feature vectors follows that of [6]. We describe the process here for completeness.

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

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

Because the current probe is placed right after the voltage regulators, other board components introduce interference to the measurements. Particularly, the RF transceiver introduces a low frequency, but strong, oscillation due to charge pumps. Because software-related power consumption is characterized by sharp transitions, the first difference between traces is used.

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

By using the sample difference, just as in the case of derivatives, sharp transitions are preserved while slow changing elements are greatly reduced. This operation is effectively a high-pass filter. Captured traces from the configuration of Normal Mode are used to create a signature, our target fingerprint. N traces are averaged to form the target signature and reduce the effects of random noise in our measurements.

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

For this paper, the process of extracting discriminatory features from the execution of code β consists of simple time-domain correlation against the target signature. The correlation, however, is performed on J partial sections of the signature and the trace, each section has a length $w = \lfloor L/J \rfloor$. This partial correlation is performed to avoid spreading potential differences in the power traces across the full trace.

The cross correlation for different sample lags, $0 \le k \le w$, of section $j = 1, 2, \ldots, J$ 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, and $\bar{d}$ and σ_d are the sample mean and standard deviation of the corresponding section in $d_\beta^{(i)}$.

In order to compensate for any clock drifts, we keep the maximum correlation values for different lags. This action reduces the dimensionality of our traces to only a sequence of J peak correlation values for every trace.

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

Under ideal conditions and with $\beta = \alpha$, $\hat{\rho}_{s_\alpha r_\beta^{(i)}}(j) = 1$ for every section j. Any deviation from the power consumption characteristics would be reflected by a reduced correlation factor. The final test statistic or discriminatory feature used in

this work to evaluate traces is the minimum peak correlation value for that specific trace

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

The random variable $x_\beta^{(i)}$ indicates the maximum deviation from the signature of instance i of code β. Using X_β, we can design appropriate detectors using different criteria depending on the statistical information we can gather from the system a priori.

In summary, after capturing traces, we 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, and make a decision whether the traces correspond to the execution of the target code.

5. RESULTS

For our Scenario 1, the average peak correlation values

$$\bar{\rho}_{s_\alpha r_\beta}(j) = \frac{1}{N}\sum_{i=1}^{N} \hat{\rho}_{s_\alpha r_\beta^{(i)}}(j)$$

from $N = 100$ execution instances are shown in Figure 5. There is a noticeable difference between the correlation with traces from the same configuration code, Normal Mode (α), and those from different configuration code, Turbo Mode (β). As expected, the average peak correlation value is higher for traces from the configuration of Normal Mode. Additive noise prevents a perfect correlation in this case.

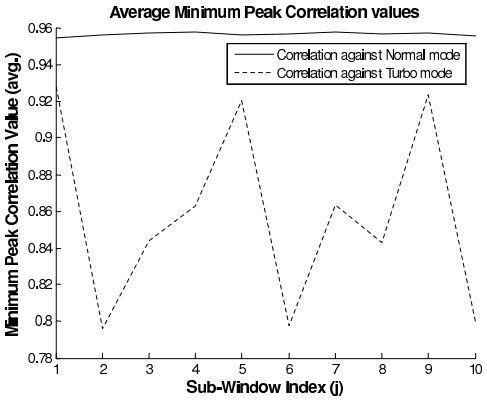

Fig. 5. Average minimum peak correlation value when original code uses default configuration

The distribution of the minimum peak correlation values for traces captured from configuration in both modes Normal and Turbo, X_α and X_β, is given in Figure 6. The separation between sample distributions clearly indicate the ability of power fingerprinting to discriminate between them. Notice that not a single sample overlaps to the opposite distribution.

Having these sample distributions, it is possible to design a suitable detector using the Neyman-Pearson criteria. The difference between distributions, however, is so large that the probability of making a classification error is negligible. The robust performance of power fingerprinting in discriminating between both configurations is expected, given that dynamic power consumption depends not only on instructions and inter-instructions transitions but also on addresses and parameters. In Scenario 1, we are greatly disrupting the execution patterns by adding extra instructions, which requires fetching from different memory locations, different parameters, and different inter instruction transitions. Hence, a marked difference is expected.

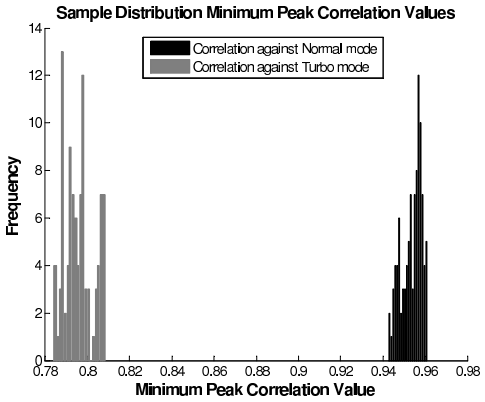

Fig. 6. Minimum peak correlation values sample distribution when original code uses default configuration

Scenario 2 is more challenging from the PF perspective. In this scenario, the Normal Mode configuration is made explicitly. Hence, the calls to write the baseband registers in the RF module are made regardless of whether the Normal or Turbo modes are configured. This time, there are no instruction or address changes, only parameters are modified. The average minimum peak correlation values of the traces captured in our second scenario are shown in Figure 7 and their sample distributions in Figure 8

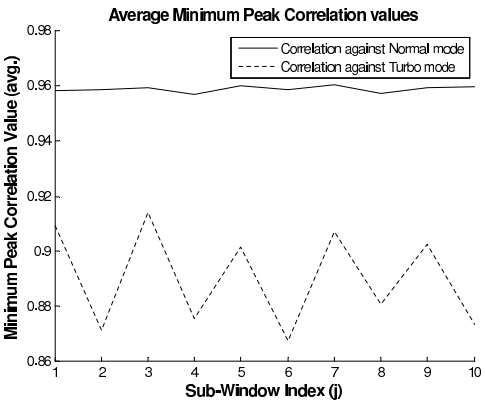

Fig. 7. Average minimum peak correlation value when original code uses explicit configuration

We can see how the distributions are closer than in the previous scenario, but still not a single trace is misclassified. The probability of a classification error using a Neyman-Pearson criteria is once again negligible.

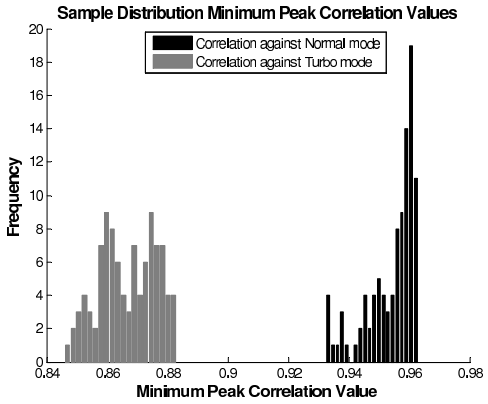

Fig. 8. Minimum peak correlation values sample distribution when original code uses explicit configuration

These two scenarios cover a large number of potential attacks. If the attacker manages to insert malware by means other than having the source code, it would be a scenario similar to our first experiment. Notice, however, that in order to detect execution deviations, it is necessary to pre-characterize the authorized code, including every execution path. This would be necessary in order to catch an attacker that manages to insert configuration code outside the configuration section.

On the other hand, if an attacker does not require to insert malicious code to configure the RF module to Turbo Mode, as in the case of the explicit configuration used in the second scenario, the approach can still detect the execution violations as demonstrated in our second experiment.

6. CONCLUSIONS

In this paper we presented the results of two feasibility experiments that further support the application of power fingerprinting in integrity assessment for regulatory purposes. We successfully demonstrated the ability of power fingerprinting to detect when the expected configuration code is modified in a simple, but real, radio platform.

In the first experiment we were able to detect when unauthorized code is inserted to change the RF module in the PICDEM Z board to Turbo Mode. Furthermore, we were also able to detect when the parameters change during an explicit configuration of the RF module to Turbo instead of Normal mode. In both experiments, we were able to detect with a 100% accuracy modifications in the configuration code that can lead to deviations in spectral emissions. While these results by themselves do not guarante the effectiveness of power fingerprinting in all platforms and all applications, they present strong evidence of its feasibility and potential impact.

References

[1] N.H.E. Weste and K. Eshraghian. *Principles of CMOS VLSI Design: A Systems Perspective*. Addison-Wesley, 2nd edition, 1993.

[2] S. E. Chodrow, F. Jahanian, and M. Donner. Run-time monitoring of real-time systems. In *Proceedings of the Twelfth Real-Time Systems Symposium*, Dec 1991.

[3] A. K. Mok and L. Guangtian. Efficient run-time monitoring of timing constraints. In *Proceedings of the Third IEEE Real-Time Technology and Applications Symposium*, Jun 1997.

[4] X. Wang, Y. Yin, and H. Yu. Finding collisions in the full sha-1. In *Proceedings of Crypto '05*, August 2005.

[5] S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007.

[6] Carlos R. Aguayo Gonzalez and Jeffrey H. Reed. Power fingerprinting in SDR and CR Integrity Assessment. In *IEEE Military Communications Conference (Milcom)*, 2009.

[7] Carlos R. Aguayo Gonzalez and Jeffrey H. Reed. Dynamic Power Consumption Monitoring in SDR and CR Regulatory Compliance. In *Proceedings of the SDR Forum Technical Conference*, December 2009.

[8] T. Laopoulos, P. Neofotistos, C. A. Kosmatopoulos, and S. Nikolaidis. Measurement of current variations for the estimation of software-related power consumption. *IEEE Transactions on Instrumentation and Measurement*, 52(4), August 2003.

[9] Microchip Website. *http://www.microchip.com/wireless*.

[10] IEEE Computer Society. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), 2003.