RASCv2: Enabling Remote Access to Side-Channels for Mission Critical and IoT Systems

YUNKAI BAI, ANDREW STERN, JUNGMIN PARK, MARK TEHRANIPOOR, and DOMENIC FORTE, University of Florida, USA

The Internet of Things (IoT) and smart devices are currently being deployed in systems such as autonomous vehicles and medical monitoring devices. The introduction of IoT devices into these systems enables network connectivity for data transfer, cloud support, and more, but can also lead to malware injection. Since many IoT devices operate in remote environments, it is also difficult to protect them from physical tampering. Conventional protection approaches rely on software. However, these can be circumvented by the moving target nature of malware or through hardware attacks. Alternatively, insertion of the internal monitoring circuits into IoT chips requires a design trade-off, balancing the requirements of the monitoring circuit and the main circuit. A very promising approach to detecting anomalous behavior in the IoT and other embedded systems is side-channel analysis. To date, however, this can be performed only before deployment due to the cost and size of side-channel setups (e.g., and oscilloscopes, probes) or by internal performance counters. Here, we introduce an external monitoring printed circuit board (PCB) named RASC to provide remote access to side-channels. RASC reduces the complete side-channel analysis system into two small PCBs (2×2 cm), providing the ability to monitor power and electromagnetic (EM) traces of the target device. Additionally, RASC can transmit data and/or alerts of anomalous activities detected to a remote host through Bluetooth. To demonstrate RASCs capabilities, we extract keys from encryption modules such as AES implemented on Arduino and FPGA boards. To illustrate RASC's defensive capabilities, we also use it to perform malware detection. RASC's success in power analysis is comparable to an oscilloscope/probe setup but is lightweight and two orders of magnitude cheaper.

CCS Concepts: • **Security and privacy → Intrusion/anomaly detection and malware mitigation;**

Additional Key Words and Phrases: Side-channel analysis, power, electromagnetic radiation, AES, buffer overflow, code injection, return-oriented program

ACM Reference format:
Yunkai Bai, Andrew Stern, Jungmin Park, Mark Tehranipoor, and Domenic Forte. 2022. RASCv2: Enabling Remote Access to Side-Channels for Mission Critical and IoT Systems. *ACM Trans. Des. Autom. Electron. Syst.* 27, 6, Article 65 (June 2022), 25 pages.
https://doi.org/10.1145/3524123

The authors thank the US Army Research Office for grant no. W911NF-19-1-0102 to support this research.
Authors' addresses: Y. Bai, A. Stern, J. Park, M. Tehranipoor, and D. Forte, 601 Gale Lemerand Dr., University of Florida, Gainesville, Florida, USA, 32611-6200; emails: baiyunkai@ufl.edu, andrew.stern@ufl.edu, jungminpark@ufl.edu, tehranipoor@ece.ufl.edu, dforte@ece.ufl.edu.

1084-4309/2022/06-ART65 $15.00
https://doi.org/10.1145/3524123

1 INTRODUCTION

The Internet of Things (IoT) has revolutionized the way we interact with technology at home and work. The IoT corresponds to the interconnection of uniquely identifiable sensors, actuators, and embedded computing devices within the existing Internet infrastructure, which enhances computing power and analytic capabilities of individual objects by linking them to additional resources and increasing the interaction between objects and their environments [48]. For example, governments and companies around the world have aggregated temperature readings or symptom data from IoT devices or smartphones to track a disease outbreak [8] or to fully automate patient-care workflow during the COVID-19 pandemic. The impact of COVID-19 is accelerating the adoption of the IoT in the healthcare sector and wearable devices [45].

However, IoT devices also face various malware attacks, including ransomware, which has been on the rise during the pandemic. Ransomware [49], a type of malware that threatens a victim's data or blocks access, initially reported in 1989, infected 20,000 disk drives of the participants of the World Health Organization's AIDS conference [32]. It can be categorized into three basic types: crypto ransomware, locker ransomware, and hybrid ransomware [49]. A crypto ransomware cracks or implements a public-private key by penetrating botnets into IoT devices. In 2008, GPCode.AK [2] was unleashed and began spreading from PC to PC, locking or encrypting the victim's files and demanding money, unless victims could crack a 1024-bit RSA key in 72 hours. Locker ransomware alters the functionalities of IoT devices and restricts user access to devices. One noteworthy example of locker ransomware was the Koler [3] ransom attack in 2014. Koler uses social engineering to trick its victims into installing the app by offering enticing adult-themed apps or fake app updates. More importantly, uninstalling Koler can be challenging because it obtains admin rights during installation. Similar to Koler ransomware, lockerpin [5] ransomware is also implemented on the smartphone. However, lockerpin ransomware directly locks a victim's system and extorts the victim for payment in virtual currency. Hybrid ransomware attacks that enable encryption and locking mechanisms are more dangerous because the device data and functionality could be compromised. In 2014, CTB-Locker [4] ransomware was discovered. It could install itself on its victims' computers and claim money when it finished encrypting victims' files.

Malware can infect a system through buffer overflow attacks, return-oriented program (ROP) attacks, and code injection attacks, which hijack the control flow of computing devices. A buffer overflow [15] occurs when a program copies a value to the buffer array that is shorter than the incoming data, overwriting the adjacent memory locations. Sometimes, buffer overflows occur when unsafe copy functions (e.g., `strcpy` in the C programming language) are used in a program. These functions can copy oversized data to the target memory without a bound check. The target memory cannot hold the entirety of the oversized data; thus, part of the data overflows and alters the adjacent memory values. An example of a buffer overflow is presented in Figure 1(a). First, the oversized data "0x44556541235654654" is copied into the target memory address. Then, part of the data is copied into the target memory and the value in the adjacent memory also gets updated.

Typically, buffer overflows occur unintentionally from uninformed programming choices. However, the buffer overflow can also be utilized by attackers. If attackers could gain access to the original code, they could substitute a state function with malicious functionality, overwriting some important values such as private keys in the RSA or AES encryption modules. The successful replacement of a secret key can cause serious security problems. One example is demonstrated in [46] in which the Code Red worm exploited a buffer overflow in Microsoft's Internet Information Services (IIS) and infected over 350,000 hosts.

The buffer overflow attack can easily be prevented by performing a bound check every time a copy occurs. On the other hand, the ROP attack does not need to rely on unsafe functions. ROP

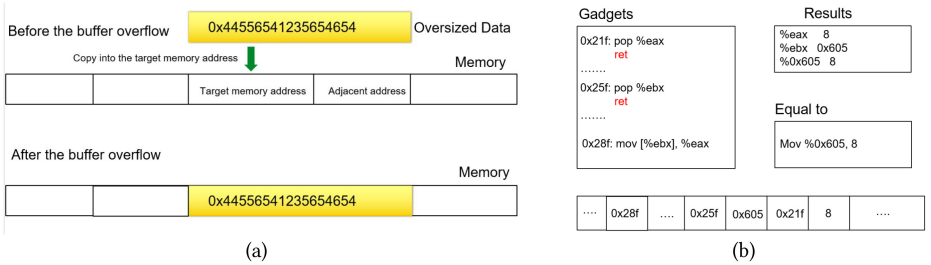

Fig. 1. (a) Buffer overflow attack and (b) ROP attack examples.

attacks utilize the existing gadgets, which end with a return instruction and are stored in the library. These gadgets perform well-defined operations, such as a `load`, and an `xor`, or a `jump`. Each gadget specifies certain values to be placed on the stack that make use of one or more sequences of instructions from libc. The attacker [36] connects useful gadgets to achieve desired operations that did not exist in the library (e.g., copying data into a secret address or printing out private keys). Figure 1(b) presents one example of the ROP attack. In Figure 1(b), the attacker aims to modify the secret value stored in the address "0x605" in the x86 system, and there is no direct command in the library to achieve this goal. Thus, the attacker exploits three separate gadgets in the addresses 0x21f, 0x25f, and 0x28f. The first and second gadgets pop value 8 and value 0x605 into the address eax and ebx (the pointer starts at value 8). The third gadget in the 0x28f copies data from value in the address eax to address ebx. With these three gadgets, the attacker could move data 8 into address 0x605.

The ROP attack assumes that the attacker could call gadgets in the library. However, in many cases, the attacker could access the original code and insert malicious code pieces directly. In many real working scenarios, such as autonomous driving and face recognition, any slight modifications to mechanical parameters could cause serious problems. For example, a car crash could quickly happen if the attacker makes changes to the threshold breaking distance in the autonomous car's code. The code injection attack also happens in cloud services. For example, in [18], hackers from APT group Team GhostShell targeted 53 universities using SQL injection, stealing and publishing 36,000 personal records belonging to students, faculty, and staff.

In addition to such malware attacks, some modifications to the hardware—such as hardware Trojans—can also cause threats to IoT devices. A hardware Trojan [42] is an intentional modification to the target circuit—such as ASICs, microprocessors, and microcontrollers—before or during fabrication. A hardware Trojan can later be activated when the target circuit meets some predefined conditions. For example, the hardware Trojan can be activated to leak a secret key from an encryption module.

To mitigate these threats, different defense mechanisms have been proposed. In general, they can be divided into software-based and hardware-based methods. The software-based methods, such as StackGuard [15] and program shepherding [26], perform control-flow integrity (CFI) checks on indirect branches. They can be added as a part of a compiler optimization step, static binary rewriting, or through dynamic library translation. StackGuard [15] is a compiler extension that enhances the executable code produced by the compiler so that it can detect and thwart buffer-overflow attacks against the stack. Program shepherding [26] prevents the execution of malicious code by monitoring all control transfers to ensure that each satisfies a given security policy. However, based on [16], the software-based CFI architectures are sometimes writable, and the runtime data structure can be modified to bypass the defend mechanism [16].

Compared with software-based methods, the hardware-based methods use monitoring circuits on the target device. There are two kinds: internal and external. The internal hardware-based method adds a monitoring circuit into the target device for detecting abnormal behaviors, such as the CFI-based monitor [13], the debug interface monitor [6], and the memory access monitor [50]. However, monitoring circuits need to modify the original circuit; therefore, they increase the overhead. Moreover, sometimes it is unrealistic and costly to adjust the original design. In contrast to the internal-based methods, the external hardware-based methods are based on the principle side-channel analysis system. They analyze the side-channel leakage, such as EM and power traces of the target device, to identify the abnormal behavior with the oscilloscope and the EM probe. For processing these side-channel traces, different algorithms are developed [30], which can be divided into the fine-grained method [19, 28, 31, 39, 40] and the coarse-grained method [14, 20, 51]. The coarse-grained methods can be applied for malware detection with repetitive features [29, 30]. Some coarse-grained method monitors [20, 51] measure the loop time of the target device. Examples include WattsupDoc [14] and EDDIE [29]. The fine-grained method is based on instruction-level granularity [30]. In [19, 28, 31, 39], the power traces are analyzed to disassemble instruction flow of the target device. In [31], Park et al. propose a disassembler based on the power-based side-channel to analyze the real-time operation of embedded systems by utilizing machine learning algorithms, Kullback-Leibler (KL) divergence, and principal component analysis (PCA). PCA is generally used for unsupervised dimensionality reduction [24]. It projects each data point onto only the first few principal components, which keeps the data variance as close as possible to the original. Experimental results demonstrate that the trained disassembler can recognize test instructions, including register names, with a high success rate with various machine learning methods: quadratic discriminant analysis (QDA) [27], linear discriminant analysis (LDA) [27], support vector machines (SVMs) [12], and neural network (NNs) [17]. The fitcdiscr function in the MATLAB statistical toolbox is used to construct LDA and QDA classifiers and the train function in the MATLAB deep learning toolbox is used to train a neural network classifier. LIBSVM [12] is used for the SVM classifier with RBF kernel. Among these four classifiers, QDA and the SVM classifier achieve a higher successful recognition rate (>99%) than the rest.

Even though the existing external hardware-based monitors have advantages in keeping the original design of the target device unchanged, they are based on the traditional side-channel analysis systems, which need expensive equipment such as oscilloscopes and EM probes. In addition, the conventional side-channel system is too large to be used to *continuously* monitor an IoT device for malware outside of the laboratory setting.

In summary, the limitations of the existing methods are apparent. Software-based methods can be bypassed, especially in hardware. Internal hardware monitoring needs to modify the original design and cannot be used in legacy devices. The existing external monitors are costly, infeasible for in-field use, and power hungry. To resolve the latter in particular, we propose RASC (short for *remote access to a side-channel platform*). RASC is an external monitor that miniaturizes the traditional side-channel system into two tiny PCB boards and provides remote communication either to perform analysis (e.g., disassembly) off-board or to communicate alerts of anomalous behavior occurring on the device being monitored. When working in ideal conditions (Figure 2), RASC could be arranged above the target device and connect to the power pin of the target circuit for the power supply. RASC gathers power traces from the power pin and EM traces with an internal near-field antenna. RASC also possesses two analog-to-digital converters (ADCs) so that power and EM traces can be collected at the same time. A Bluetooth module on RASC can perform wireless communication to base stations. While Bluetooth itself cannot communicate over large distances, its signal can be collected by a nearby wireless link to do so. This allows each RASC

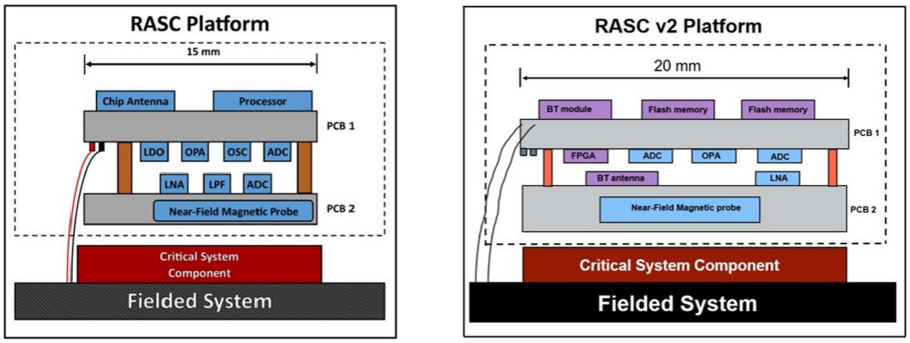

Fig. 2. (a) RASC version 1 (v1) [38] and (b) RASC version 2 (v2) [this article]; the purple chips in v2 are added or modified from v1.

device to be cheap and consume less energy while such a wireless link can collect and transmit the signals from multiple RASC devices.

In this article, the following are our main contributions.

(1) RASCv2's design is described and its features are demonstrated, including power measurement, EM measurement, on-board processing, and remote communication.
(2) RASCv2 is compared with other side-channel measurement systems, including RASCv1, traditional oscilloscope and probe, and Chipwhisperer [41]. Compared with RASCv1, RASCv2 upgrades its functionality in sampling speed, EM antenna design, and Bluetooth communication. Unlike the oscilloscope and the Chipwhisperer, RASCv2 has advantages in low cost and tiny size while obtaining similar performance.
(3) RASCv2's offensive capabilities are established by using its power and EM measurements in a CPA attack to extract AES subkeys from different architectures (Arduino microcontroller and field-programmable gate array [FPGA]). The success rate of AES subkey extraction in Arduino UNO with traces from RASCv2 is close to the result from the oscilloscope and is proven to be stable across Arduino UNOs.
(4) To demonstrate RASCv2's defensive capabilities, we successfully train a spectral profiling-based SVM classifier with power traces collected by RASCv2, and achieve high detection rates for recognizing three kinds of malware attacks inserted into 10 different benchmarks.

The rest of the article is organized as follows. Section 2 discusses related work, including the first version of RASC and other hardware monitoring systems. Section 3 provides discussion of RASCv2's hardware, comparison between RASCv2 and other side-channel monitoring systems, and security characteristic of RASCv2. Section 4 introduces two algorithms adopted in the AES-128 subkey extraction and malware detection experiments. We also explain the experimental setup in Section 4. Section 5 demonstrates the attack capability of RASCv2 against the AES-128 module implemented on an FPGA and an Arduino UNO. Section 6 presents high successful detection rates toward different attack scenarios (buffer overflow attack, code injection attack, and ROP attack) in 10 different benchmarks using trained SVM classifiers with power traces collected by RASCv2. Section 7 presents our conclusions and discusses future directions for RASC.

2 RELATED WORK

Many side-channel external monitoring methods have been proposed for detecting malware. One efficient way for detecting malware is to validate the control flow integrity [30] of the target device.

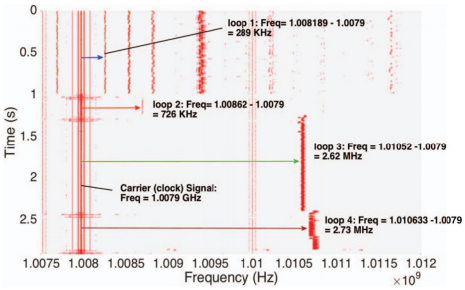

Fig. 3. Spectrogram of benchmark "Basicmath" in [34].

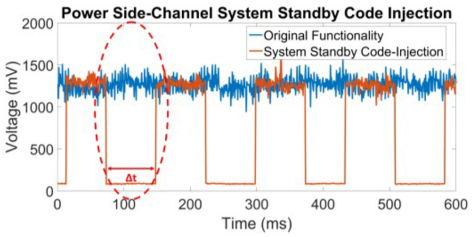

Fig. 4. Visual proof-of-concept for code-injection attack detection using RASCv1 [38].

As introduced before, these CFI techniques could be classified into coarse-grained CFI methods and fine-grained methods based on the granularity of monitored activities. Some related works in the malware detection area are introduced here.

WattsUpDoc. Clark et al. [14] proposed WattsUpDoc for detecting malware attacks on an embedded medical device. WattsUpDoc [14] collects power traces at runtime and uses supervised machine learning (ML) to identify abnormal activities of the medical device. The output of the supervised ML algorithm is a functional state such as idle, booting, and shutdown of the medical device. The algorithms could achieve 94% detection rate toward known attacks and at least 85% accuracy for unknown attacks. Compared with WattsUpDoc, this article (RASC) not only focuses on the power channel but also focuses on the EM channel. Further, WattsUpDoc is applied at the outlet rather than near the target device.

EDDIE [29]. Nazari et al. proposed EDDIE to detect abnormal behaviors caused by code injection attacks in program execution with EM traces. EDDIE relies on spikes in the EM spectrum due to the periodic activity in the monitored execution. This involves two phases: the training phase, followed by the monitoring phase. In the training phase, EDDIE is trained under the normal execution behavior in terms of peaks in the EM spectrum. In the monitoring phase, EDDIE identifies peaks in the observed EM spectrum and compares these peaks to those learned during training. EDDIE achieves at least 92% accuracy against code injection attacks. Compared with RASC, EDDIE relies on a commercial antenna, uses an oscilloscope, and does not perform wireless communication, making it more expensive and difficult to perform remotely.

Syndrome. Syndrome uses spectral profiling [35] to analyze program execution using spectral "spikes"produced by periodic program activity with EM traces. The EM traces are transformed into the time and frequency domain using the short-time Fourier transform, and the frequency and per-iteration execution time of the loop are analyzed. Figure 3 presents the spectrogram of the benchmark "Basicmath" from Mibench [21]. In benchmark Basicmath, four loops are included, all of which can be clearly visualized in the spectrogram. The spectrogram also brings more details about frequency and the time period of the internal loop in Basicmath. The frequency and per-iteration execution time matches the core frequency of the processor and the number of loops. Any code injections to the main program can be detected by comparing differences between malware spectrograms and the malware-free spectrogram. For developing malware attacks, Sehatbakhsh et al. propose Syndrome [33] to detect abnormal activities on medical IoT devices based on [35]. Sehatbakhsh et al. trained all of the important spikes in the spectrum, which includes features corresponding to the internal loop. In the monitoring phase, the K-S test is adopted to compare the spectrum of the gathered testing EM trace and the trained spectrum in the training phase.

Abnormal behavior is decided by the deviation between the testing spectrum and the trained spectrum. Syndrome achieves a 100% true-positive detection rate against control flow attacks on different devices, including Arduino, Nios-II, TS, and Olimex. Like Syndrome, RASC also employs spectral profiling for malware detection.

NIPAD. Xiao et al. propose NIPAD [25] to detect abnormal activities in a programmable logic controller (PLC). Unlike EDDIE [29] and Syndrome [33], NIPAD [25] collects power traces and extracts a discriminative feature set. Later, the features of normal samples are trained based on a long short-term memory (LSTM) neural network. Abnormal behavior could be identified by comparing the predicted sample and the actual sample. The experiment results show a high successful detection rate of LSTM networks (90.33%) even if only one line of the original program is modified. In addition, NIPAD [25] compares three different methods that could be used for malware detection: a one-class SVM classifier, LSTM network, and correlation-based algorithm. LSTM achieves a higher successful recognition rate against modification and lower equal error rate than the other two methods. In this article, we use power and EM, spectral profiling, and a one-class SVM for malware detection.

Bridges et al. [11] present an experimental design and algorithm for power-based malware detection on general-purpose computers. For detecting abnormal activities, different features of the power trace are calculated and trained into a classifier for identifying malware attacks. Bridges et al. [11] also compare different kinds of features, including mean, variance, skewness, and so forth. The authors found that the proposed one-class method outperforms supervised learning with kernel SVMs.

In summary, these works are all solutions for side-channel external monitoring. Some [29, 33, 35] target EM traces and others [11, 14, 25] analyze power traces. The most significant differences between these and our proposed work are related to the improved methods to collect EM/power traces by RASC. These methods need some traditional side-channel analysis system, such as the oscilloscope and the EM probe, to collect power/EM traces. The large size and high cost limit the range of usage. However, the tiny body and cheap cost of RASC allow us to fit it into some narrow place. In addition, RASC supports remote communication so that we could fabricate large quantities of RASC, place them in different places, and monitor distributed IoTs remotely. The articles discussed here target only one channel: EM or power. However, RASC can combine EM and power traces to improve the result.

RASCv1 [38]. The original RASC builds on the principle of side-channel analysis. Side-channel analysis gathers unintentional leakages from an electronic system and processes these leakages for reverse-engineering instructions, analyzing control flow of the target device, and so forth. A predefined condition of side-channel analysis is that the device under test could be accessed physically. However, in most IoT and cyberphysical system scenarios, this condition cannot be satisfied. Hence, a remote access embedded system, termed RASC, was initially designed by Stern et al. to achieve side-channel utilization in physically inaccessible devices. As shown Figure 2(a), RASCv1 consists of two 15 × 15 mm PCB boards (denoted PCB1 and PCB2). PCB1 houses the main microcontroller, power side-channel measurement circuitry, and power regulators. PCB2 contains an internal near-field magnetic probe, an amplifier, and an ADC circuit. During in-field use, the two PCBs can stack on top of each other to perform both EM and power analysis. In [38], RASCv1 could visually identify code injection attacks. In Figure 4, power traces collected by RASCv1 show differences between original code and the code that is injected. The experiment results of RASCv1 demonstrate its potential applications, including monitoring of cyberphysical systems, real-time CPS monitoring, and secure firmware upgrading. However, the RASCv1 had shortcomings in its design. First, the response of the EM probe in RASCv1 was too low, which decreased the

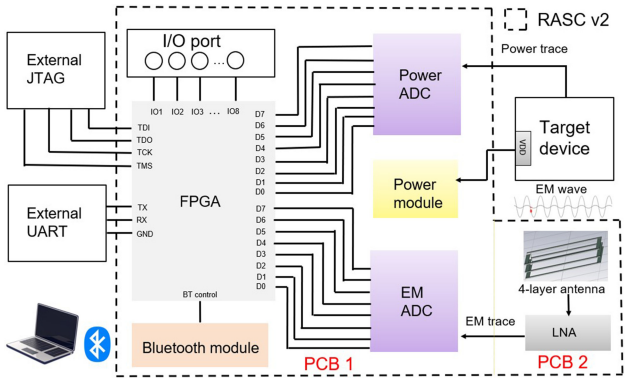

Fig. 5. RASCv2 schematic.

signal-to-noise ratio (SNR) so that RASCv1 could not meet the basic demands of EM analysis. Second, the sampling speed of the two ADCs on the RASCv1 was less than 10 MS/s, which is slower than the main clock of most target devices. For example, high-speed microprocessors such as the x86 processor are faster than 1 GHz. Even low-speed microcontroller units (MCUs) such as Arduino UNO have 16-MHz clock frequency. Thus, RASCv1 could not sample even one point for each instruction. Hence, RASCv1 could not perform key extraction or support more advanced/automated malware detection methods such as spectral profiling [35]. Third, the Bluetooth module of RASCv1 could not transmit large amounts of data due to the insufficient memory inside the Bluetooth module, thereby limiting its remote communication. In RASCv2, hardware upgrades have been made to solve these three problems; thus, RASCv2 has improved significantly compared with RASCv1.

3 OVERVIEW OF RASCV2

3.1 Structure of RASCv2

Figure 2(b) presents the structure of RASCv2. The schematic is presented in Figure 5. Similar to RASCv1, RASCv2 consists of two boards but with a larger size. The first board (PCB1) contains two ADCs for digitalizing EM and power traces, a Bluetooth module for remote communication, and a Xilinx Spartan-3E (XC3S500E) [47] FPGA for data processing. The ADC on the PCB1 is TI ADC08200 [44]. Its maximum sampling speed could be 220 MS/s. SESUB-PAN 14580 [47] is selected as the Bluetooth module since it has a tiny body (8 ×8mm) and a long range of communication (>20 m). The key reason for choosing Spartan-3E is its tiny body and large number of I/O ports. The second board (PCB2) has an internal 4-loop magnetic probe. The method connecting two boards is similar to RASCv1 in Figure 2, that is, the two boards could be stacked on top of and attached to each other. In the working scenario, RASCv1 and RASCv2 could be arranged on the top of the target chip. EM traces could be gathered by the magnetic probe and the power of the RASCv2 could be supplied by the device under test (DUT).

Compared with RASCv1, RASCv2 upgrades its functionality by selecting different components so that it can handle more complicated experiments and practical applications. More details are provided here.

(1) **ADC.** The sampling speed of ADC in RASCv2 is much faster than before. The maximum sampling speed could reach 220 MS/s. However, in RASCv1, the ADC is an internal module inside the MCU, which is slower than 10 MS/s. A target device in this article and [38] is Arduino UNO; its core frequency is 16 MHz. Clearly, a faster sampling speed of the ADC could support a more accurate analysis method.

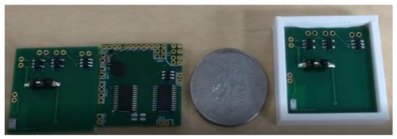

Fig. 6. (Left and center) Comparison between RASCv2 and a quarter; (Right) RASCv2 PCB2 in sample holder.

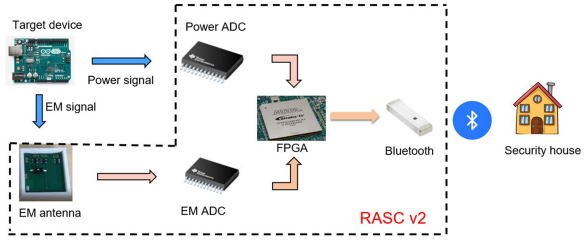

Fig. 7. RASCv2 functional diagram. RASC collects power and EM traces, can process them on its FPGA, and can transmit traces or other results to a security administrator using Bluetooth.

(2) **FPGA.** Unlike choosing an MCU in the first version, a Spartan-3E is adopted for controlling more components and providing sufficient I/O ports on the board for connecting to possible external modules such as a universal asynchronous receiver/transmitter (UART) module for test and debugging.
(3) **Amplifier and EM antenna.** In the first version, the low SNR of the EM probe limited its functionality to gather pure EM traces. Thus, in this version, we design more loops of antenna and choose better LNA components to receive correct EM signals. The correctness of the EM traces have been verified in the AES-128 key extraction experiments.
(4) **Size and Sample Holder.** In Figure 6, the size of RASCv2 is small and close to a US quarter. However, the size is still 20% larger compared with the first version. Moreover, we also design a special sample holder (shown on the right of Figure 6) for containing PCB2, which allows us to carefully and reliably position it on top of the target board.

3.2 Functionality of the Second Version of RASC

In the functional diagram (Figure 7), RASCv2 connects to the power pin of the target device for power supply and gathers power traces at the same time. The magnetic probe collects EM traces and the EM traces are sent to PCB1. When power and EM traces are received in PCB1, two ADCs digitalize the signals and the FPGA can store them in its memory. After processing data inside the FPGA, RASC can send information to security administrators for remote analysis as needed.

Compared with RASCv1, RASCv2 upgrades its functionality in terms of faster sampling speed, better EM signal, and better Bluetooth communication.

(1) **Sampling speed.** The low sampling speed of the first version limits its ability to detect minor differences caused by malware attacks. The first version can detect only huge differences happening in the target device, such as the updates of the system. In this version, the collected power traces could find differences in some parameters in the main program. This has been verified in the experiments section.
(2) **EM signal.** The response of the first-version EM probe is overwhelmed by white noise. Thus, there is no EM experiments included in the first RASC paper [38]. In this article, this problem is improved, and the EM trace can be used for extracting AES-128 subkeys.
(3) **Bluetooth communication.** The choice of the Bluetooth module in this article is SESUB-PAN-DA14580 from Dialog Semiconductor (TDK). Compared with the first version, the new Bluetooth module has advantages in detection range and a larger memory. Here, we design a simple experiment of transmitting collected power traces to show the Bluetooth module's abilities.

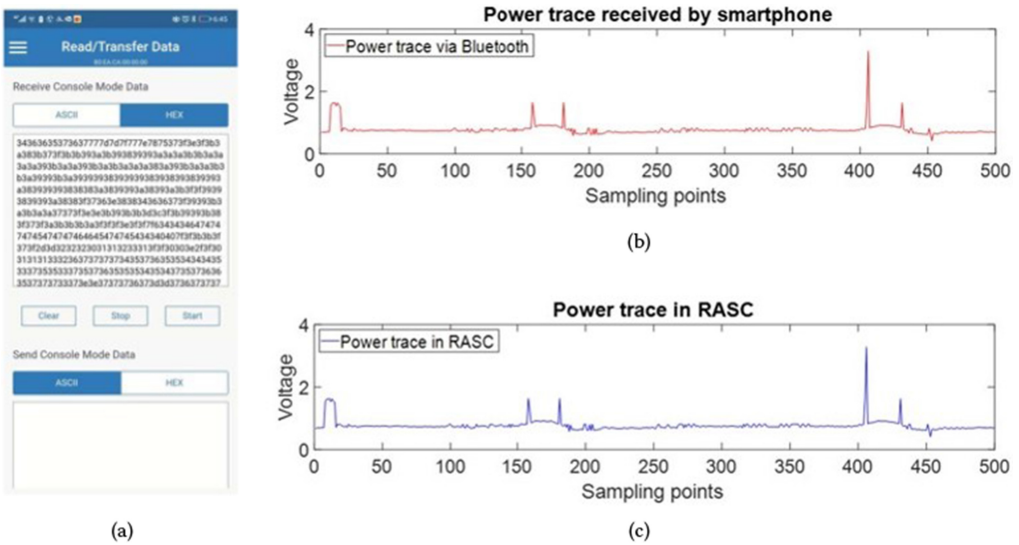

Fig. 8. Bluetooth experiment results: (a) DSP app screenshot; (b) Power trace received by smartphone; (c) Power trace sent from RASC.

Figure 8 is the screenshot of hex data received by our smartphone and the comparison between power traces stored in RASCv2 and received power traces. Figure 8(a) shows the screenshot of the DSP app from TDK company. Every two digits are hex values of its corresponding point in the power trace. For example, the first data value is 34, and the hex data 34 stands for 52 in the decimal system. Thus, the voltage value of the first point should be $\frac{52}{256} \times 3.3V = 0.6703V$. Figure 8(b) is the power trace received by the smartphone and Figure 8(c) is the power trace stored in RASC. The high similarity between the two power traces shows that the correct communication could be built between RASCv2 and security administrators. In this edition, we use RASC to transmit signals to a PC for off-board analysis. In the next version, we want RASC to perform processing internally and transmit cracked keys, disassembled instructions, and/or resulting alerts to remote computers instead.

3.3 Comparison between RASC and Oscilloscope

Table 1 shows the comparison between RASCv2, an oscilloscope, and the ChipWhisperer [41], which is a popular commercial board.

(1) **Price.** The traditional side-channel analysis system consists of a commercial EM probe and oscilloscope. The oscilloscope (Tektronix MDO 3102) we use in our lab can sample data over 5 GS/s and costs over $16,000. The commercial EM probe also costs over $200. However, the total cost of RASC is around $300 (produced at low volume), which is comparable to ChipWhisper lite edition (produced at high volume). More importantly, the price of RASC could be dropped down to a lower value if RASC is fabricated in larger quantities, like a commercial chip.
(2) **Size.** The tiny body of RASCv2 is shown in Figure 6. The tiny body could let RASC be more easily placed into narrower spaces to monitor IoT devices.
(3) **Voltage testing range.** The voltage testing range of two ADCs on RASCv2 is from 0 V to 3.3 V. For the power ADC, this range is acceptable since this range covers the power supply of most commercial FPGA/MCU development boards. However, it is not an acceptable

Table 1. Comparison between Traditional Side-channel Analysis Systems and RASCv2

List	Oscilloscope+EM antenna	RASCv2	ChipWhisperer-Lite 32-Bit
Cost	$16,000	$300	$250
Size	Large	Small	Medium
Test voltage range	Large	Small	Medium
Sampling speed	5 GS/s	128 MS/s	105 MS/s
Remote communication	No	Yes	No
Resolution	High	Low	Medium
Programmable	No	Yes	Yes

solution to the EM trace. The response of the EM antenna is around 0 V; thus, half of the EM traces read by ADC is 0 V (see Figure 10). Compared with RASC and ChipWhisperer, the oscilloscope has a much more extensive testing range. Nevertheless, this can be cheaply remedied in the next version of RASC.

(4) **Sampling speed.** Obviously, the oscilloscope has great advantages in sampling speed. As for Tektronix MDO 3054, the maximum sampling speed could reach 5 GS/s. The sampling speeds of RASCv2 and ChipWhisperer are lower than the oscilloscope. In the malware detection area, sampling speed is an important parameter. It determines the choice of target device and the accuracy of the detection method.

(5) **Remote communication.** The RASCv2 supports Bluetooth communication over 20 meters. The combination of remote communication and its tiny size could let RASC work in a narrow space remotely. ChipWhisper and the oscilloscope do not include this feature.

(6) **Resolution.** The ADC on the RASC board is ADC08200 (8 bit). The resolution of the ADC is 1.2 mV. It is clear that the oscilloscope and ChipWhisper have great advantages in detecting a faint difference in traces. Still, in our experiments, RASC performs quite well.

(7) **Programmable.** On RASC PCB1, there are 14 I/O ports. We can load programs to the FPGA on PCB1 to achieve much different functionalities. The oscilloscope can access MATLAB on the laptop and supports API code. However, it is not flexible compared with the usage of FPGA.

In conclusion, RASC has advantages in low price, small size, and remote communication. At the same time, it has limitations in the sampling speed, the resolution of the amplitude, and the voltage testing range, some of which will be improved in the next version. However, in many scenarios, such as cracking AES-128 subkeys, there is no need to sample traces at high speeds, such as 5 GS/s. In other words, RASC can serve a substitute for an oscilloscope in practical defense/offense applications.

3.4 Security Characteristics of RASC

(1) **Fabrication.** The hardware of RASC is trustworthy as compared to the DUT, which might contain hardware Trojans or malware. RASC uses well-known commercial chips (FPGAs, amplifiers, and so on) obtained from authorized distributors and its overall design is not complex. Thus, the source of components is reliable and the fabrication/integration of RASC boards can be performed in a trusted environment.

(2) **Resistance to Physical Tampering.** When operating in the field, an attacker might try to tamper with the RASC boards, components, interconnects, and more. This can be prevented by using secure enclosures or tamper-proof coatings [23, 37].

(3) **Firmware Tampering.** RASC can employ a secure boot/debugger to authenticate its own VHDL code. Thus, even when RASC is in a hostile environment, it can be protected. For example, hackers could reload a malicious program into RASC or replace its VHDL code

held in its flash memory. A secure boot would prevent any possible modifications to the codes inside RASC by verifying the signatures of RASC whenever it powers up. Moreover, the system of RASC could be locked by a security debugger to prevent the leakage of code.

(4) **Communication.** RASC sends secret data to a security house when working remotely. The communication data sent from RASC could be encrypted by an internal AES block or stream cipher block. Thus, the confidentiality of its communications can be protected. Meanwhile, the integrity and/or authentication of data could be provided by using a hash function and/or HMAC.

(5) **Removal.** If RASC is in constant communication with security administrators, it would be difficult for an attacker to remove RASC from the DUT without being detected. Doing so would stop RASC from measuring side channels from the DUT or break the communication with security administrators, which will trigger anomalies that are easy to detect by RASC or by the security house.

In short, the design and fabrication of RASC is cost-effective and these security mechanisms could be easily achieved. Meanwhile, the DUT is a chip/system likely manufactured in an advanced technology node offshore. The overseas fabrication of advanced technology nodes increases the risk of hardware Trojans. Conventional test approaches are likely to miss these hardware Trojans, which are triggered under very rare conditions. Any firmware updates to the DUT could introduce malware and the DUT, especially legacy devices, might not contain internal checks for malware, secure boots, and the like. Thus, RASC is an effective choice as a root of trust to monitor the DUT.

4 EXPERIMENTAL SETUP AND ALGORITHMS

4.1 Setup

The experimental setup for collecting power traces using RASC and an oscilloscope is shown in Figure 9. For collecting power traces using RASC, PCB1 is directly connected to pin 8 of the ATmega 328P chip. For the EM traces using RASC, PCB2 is arranged on top of the ATmega 328P chip using the sample holder. For comparison, the channel 1 of the oscilloscope MDO3102 [43] is used for collecting power traces and the channel 2 is used for collecting EM traces with a commercial EM probe. Details are found in the square insets of Figure 9.

4.2 Algorithms

There are two different experiments included in this article. The first experiment is an illustration of RASC's offensive capabilities in which it is used to extract AES-128 subkeys by correlation power analysis (CPA) of traces. The second experiment involves malware detection, in which we train a spectral profiling-based SVM classifier. The algorithms used are described next for replication by other research groups.

4.2.1 AES-128 CPA Attack Algorithms. The details of the CPA attack are explained in Algorithm 1. First, power traces (**T**) and associated plaintext (**P**) are collected by RASC and used to calculate the hypothesis data (h) for different key indexes. The index (i) here means the position of the subkey in the whole key. The 128-bit key can be divided into 16 subkeys and each subkey has 8 bits. Therefore, each subkey has 256 possible values. Thus, the hypothesis should be calculated 256 times, that is, once for each possibility. The hypothesis model that we use for the Arduino UNO is the Hamming weight (HW) model (**line 4**). The guessing value (j) of the subkey needs to be XORed with its corresponding 8 digits in the plaintext ($p(k, i)$). The result is sent to its related SBox. The HW model is based on how many bits with value of '1' exist in the result. The hypothesis model for the FPGA is the Hamming distance (HD) model (**line 6**). Here, we calculate the HD between ciphertexts and the intermediate result before the last round of encryption based on

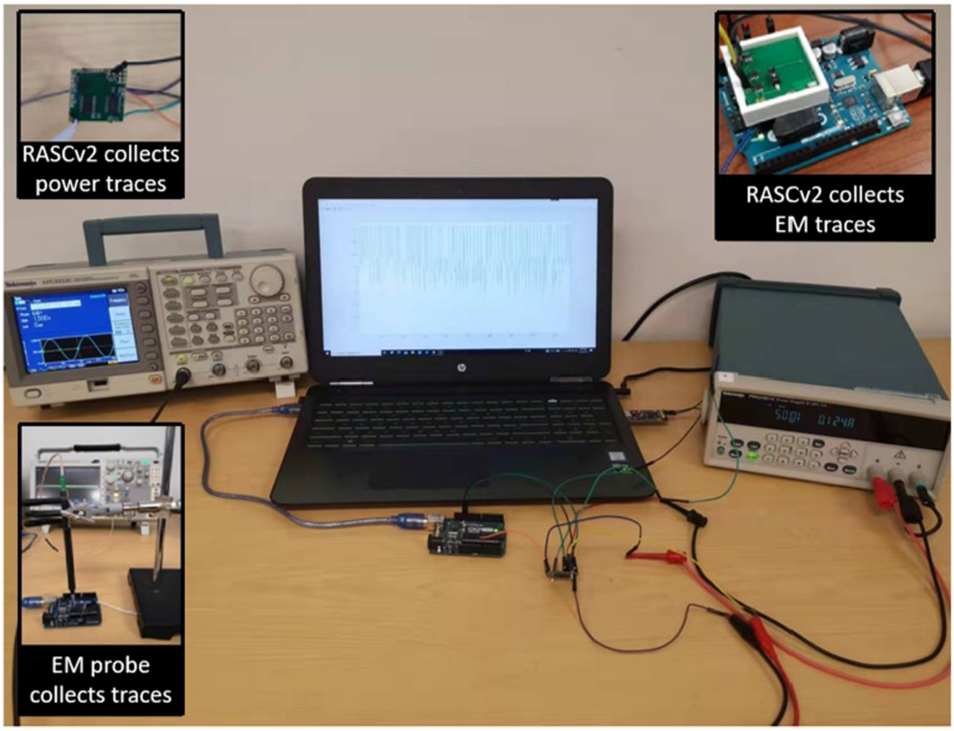

Fig. 9. Experimental setup to collect power and EM traces from ATmega 328P using RASCv2 and oscilloscope.

256 guess values. After getting the hypothesis value, the correlation coefficient can be worked out between the sampling trace points ($\mathbf{t_k}$) and the hypothesis value ($\mathbf{h_j}$) for every possibility (**line 12**). After finishing the calculation, the index value of the largest peak in the correlation coefficient is chosen as the guessed subkey (**line 15**).

4.2.2 Malware Detection Algorithms. The algorithm used for malware detection is shown in Algorithm 2. It consists of two phases: training and testing.

(1) **Training phase.** First, RASC collects N_t traces (**T**) from the target device, which runs malware-free code. Then, each collected power trace is transformed into the time-frequency domain by short-time Fourier transform (STFT) (**line 2**) and a complex matrix (**Spec**) correlated with time and frequency is obtained. *Mrow* (**line 3**) is the max row number of the matrix (**Spec**), and *Mcolumn* (**line 4**) is the max column number. Next, each complex value (*Spec(r,c)*) in a column of the matrix is transformed into its magnitude value, squared, and added to generate a value, named *Vspec1* (**line 10**). The $Mrow \times Mcolumn$ matrix (*Spec*) is then transformed into a temporary vector of size *Mrow* (*Vspec1*); this vector is stored in **Vspec** (**line 12**). We next randomly choose two vectors in **Vspec** (**line 15 and 16**), calculate the distance between these two vectors (**line 17**), and repeat it N_l times. The N_l distances (*D*) between these vectors are trained to generate a one-class SVM classifier (*S*) in **line 19**; 5% of the observations are outliers. The purpose of the trained SVM classifier is to compare the distance within malware-free traces and the distance between malware-free traces and malware traces. Obviously, the former distance is shorter than the latter distance. Therefore, the latter one will be detected as an anomaly by the trained one-class SVM classifier.

ALGORITHM 1: CPA Attack

Input: **D** : Device, Arduion UNO or FPGA
Input: **T** : power trace, **P** : plaintext, **C** : ciphertext, i : index
Output: k_i : the i^{th} guess subkey

```
for k=0:N_t − 1 do
    for j=0:255 do
        if (D == Arduino UNO) then
            h(k, j) = HW(SBox(p(k, i) ⊕ j))
        else if (D == FPGA) then
            h(k, j) = HD(c(k, i), Inv_SBox(c(k, i′) ⊕ j))
        end
    end
end
for k=0:N_s − 1 do
    for j=0:255 do
        r(k, j) = correcoef(t_k, h_j)
    end
end
k_i = arg max_j r(k, j)
```

(2) **Testing phase.** In the testing phase, a testing trace (**T'**) is collected from the target device running malware-free code or malware-inserted code. The testing trace is transformed into the testing vector just like the training phase (**lines 2–line 12**). We calculate the distance between this testing vector (*Dtest*) and average them (*A*) in **line 33**. The one-class SVM classifier (*S*) gives a score (*Score*) to these vectors (**line 34**). If the score is larger than 0, it means that it is within the separating hyperplane of the classifier; thus, the testing trace is classified as a *malware-free trace* (**line 36**). If the score is less than 0, the testing trace is considered a *malware-infected trace* (**line 38**).

5 AES-128 KEY EXTRACTION EXPERIMENTAL RESULTS AND DISCUSSION

In this experiment, two different target boards are used for performing SCA attacks to extract an AES-128 secret key. The first is the Arduino UNO based on an AVR 8-bit microcontroller (ATMega 328P), and the second is the SAKURA-G FPGA board [1].

5.1 Attack on Arduino UNO

The flow of this experiment is close to the description in the functional diagram. The plaintext is sent to the Arduino UNO and RASC starts to collect the power/EM traces when the Arduino UNO is encrypting the data using AES. The power/EM trace has 7,000 points; these traces can be sent to the laptop through UART. After collecting plaintext, ciphertext, and the corresponding power/EM traces, the extraction process using MATLAB occurs as described in Algorithm 1. Figure 10 presents the comparison between collected EM and power traces by RASCv2. Some points are missing in the EM trace due to the insufficient voltage detection range of the EM ADC. The voltage-detecting range of the EM ADC is from 0 V to 3.3 V. However, some parts of the EM signal are below 0; sthus, these portions of traces are missing (see Figure 10(b)).

Before cracking subkeys, the t-test result for RASC and oscilloscope is performed to verify that the power trace from the Arduino could be attacked. For the t-test experiment, we perform a random-versus-fixed test vector leakage assessment (TVLA) test [9]. For the random dataset, the

ALGORITHM 2: Malware Detection Algorithm

```
Input: T - Power traces for training, T' - Power trace for testing
Output: Malware-free trace or Malware trace
for i=1:N_t do
    Spec = STFT(T(i)))
    Mrow = length(Spec)
    Mcolumn = width(Spec)
    for r=1:Mrow do
        tmp = 0
        for c=1:Mcolumn do
            tmp = tmp + Spec(r,c)*Spec(r,c)
        end
        Vpec1(r) = tmp
    end
    Vspec(i) = Vspec1
end
for i=1:N_l do
    R1 = random[1, N_t]
    R2 = random[1, N_t]
    D(i) = Distance(Vspec(R1),Vspec(R2))
end
S = SVM(D,outlier=0.05)
SpecT' = STFT(T')
MrowT' = length(SpecT')
McolumnT' = width(SpecT')
for r=1:MrowT' do
    tmp = 0
    for c=1:McolumnT' do
        tmp = tmp + SpecT'(r,c)*SpecT'(r,c)
    end
    VspecT'(r) = tmp
end
for i=1:N_t do
    Dtest(i) = Distance(VspecT',Vspec(i))
end
A = mean(Dtest)
Score = predict(A,S)
if Score>0 then
    T' is Malware-free trace
else
    T' is Malware trace
end
```

first plaintext is 0xAAAAAAAAAAAAAAAA. Then, the ciphertext is used as the next plaintext. For the fixed dataset, the fixed plaintext is fixed to 0xAAAAAAAAAAAAAAAA. After gathering all of the traces, we use the formula described in [9] to calculate the t-test result. Note that the oscilloscope and the RASC share the same trigger signal to keep with their start time. The

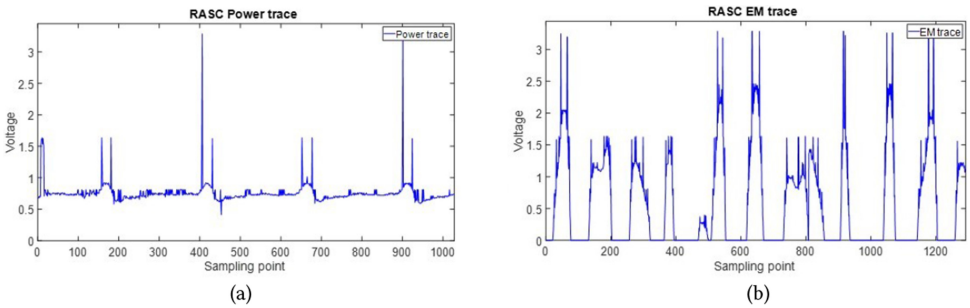

Fig. 10. (a) RASC's power trace versus (b) EM trace.

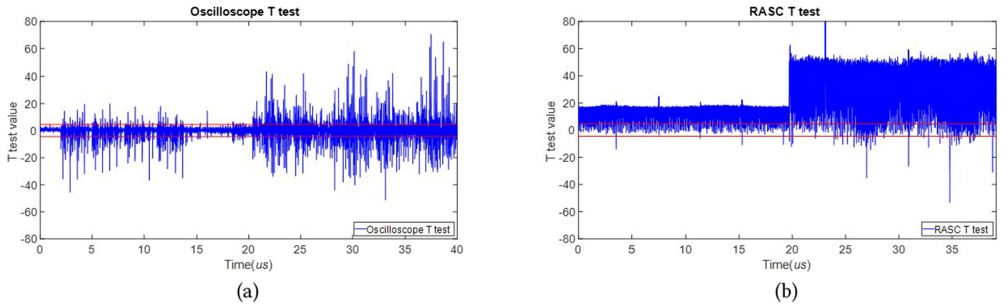

Fig. 11. (a) Oscilloscope t-test versus (b) RASC t-test.

comparison is presented in Figure 11. Both oscilloscope and RASC t-test results are beyond the threshold (< -4.5 and > 4.5), which means that the power traces of the Arduino are breakable. More importantly, the similarity between two graphs shows correctness of the collected power traces from RASC. In Figure 11(a), the t-test value becomes larger after 2,000 sampling points. The time period after 2,000 sampling points is more easily attacked compared with the period before 2,000 sampling points. Similarly, the same situation happens for RASCv2 after 2,550 sampling points in Figure 11(b). The different start times of the crack time periods in Figures 11(a) and 11(b) are caused by different sampling speeds of the RASC (128 MS/s) and the oscilloscope (100 MS/s). Since the start time is the same, the start time point of the crack time period in the oscilloscope experiment is $2{,}000 \times \frac{1}{100M} = 2\mu s$. For RASC, the start time point of the cracking time period is $2{,}550 \times \frac{1}{128M} = 1.992\mu s$, which is close to the result from the oscilloscope.

After this verification, the power traces and EM traces from RASC and the oscilloscope were used to extract AES subkeys. For analyzing the successful detection rate, we collect plaintexts and associated power traces randomly from the dataset and repeat the key extraction 50 times. In Figure 12(a), we compare the successful extracting rate of subkey #6 between power traces collected by RASC versus oscilloscope as the number of traces increases. This figure also includes a comparison of the success rate between power traces collected by RASC from three different Arduino UNOs. The success rate of RASC is very close to the oscilloscope in the power channel. It proves the correctness of the power traces from RASC. Moreover, the stability of RASC collecting power traces is proved since the successful cracking rate of power traces from different Arduino UNOs are close to each other.

In Figure 13, the comparison of cracking subkey #6 using EM traces collected from RASC and from the commercial EM probe and oscilloscope are presented. The success rate of the oscilloscope

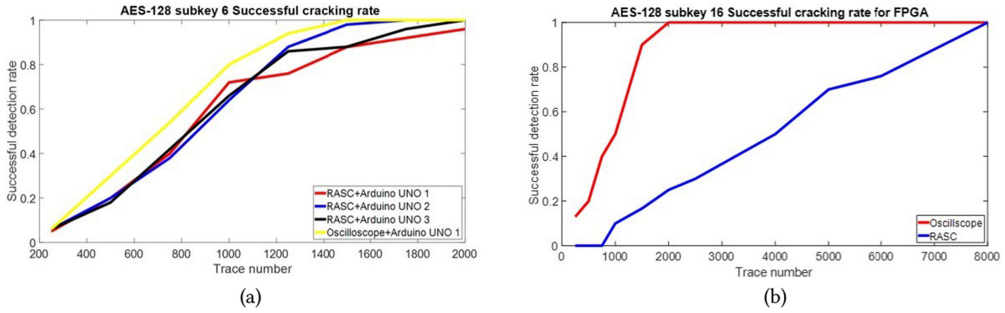

Fig. 12. (a) AES-128 subkey #6 successful cracking rate for RASC. (b) AES-128 subkey #16 successful cracking rate for FPGA.

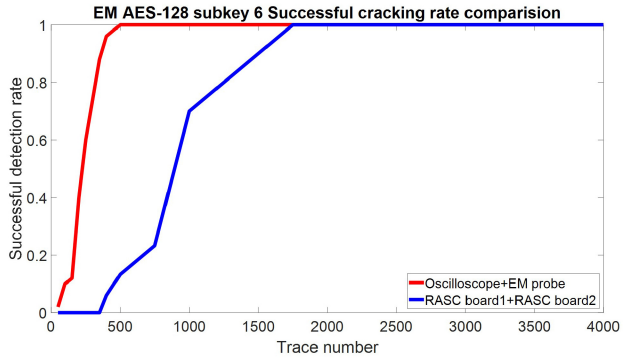

Fig. 13. AES-128 subkey #6 successful cracking rate for EM channel and RASCv2 versus oscilloscope MDO3102.

+ probe is much better than the result from the RASC. The first reason is the difference between the internal antenna inside RASC PCB2 and the commercial EM probe. The second reason is insufficient detecting range of EM ADC so that RASC misses some pieces of data (shown previously in Figure 10(b)). However, the result still demonstrates that RASCv2 can crack subkeys even if it misses almost half of the data. If the EM detection range of RASC improves in its next version, we expect success rates comparable to the oscilloscope.

5.2 Attack on SAKURA-G Board

The second target is the SAKURA-G board [1], which is a commercial FPGA board made for side-channel analysis experiments, including a controller FPGA and a main FPGA. We use all source codes for the AES implementation on an FPGA and software called SAKURA checker for generating different plaintexts supported by [1]. The experiment setup is similar to the Arduino AES-128 subkey extraction experiment.

Unlike Arduino UNO, the hypothesis model for attacking FPGA is the HD model. The HW model is not suitable here due to the different architecture of the FPGA. To be more specific, unlike precharging in the microcontroller circuit, the FPGA processes data in parallel and stores data into the data register after every round. Thus, calculating the HD between data in the data register before and after one round is the more effective approach. The key reason is that more bits change in the data register, causing more power consumption. As with many other papers [7, 10, 22], we

choose to attack the final round of AES encryption. The HD is calculated between ciphertext and the intermediate result before coming into the last round.

In the end, with 13,000 power traces, 14 subkeys can be successfully cracked. For analyzing the success rate, we collect plaintexts and associated power traces randomly from the dataset and repeat the key extraction 50 times. The comparison of subkey #16 key extraction success rate is presented in Figure 12(b). Compared with Figure 12(a), for cracking keys from FPGA, the oscilloscope outperforms RASC. RASC needs at least 8,000 traces to achieve a 100% successful detection rate. However, the oscilloscope needs only 2,000 traces. We think the reason is the different requirements of accuracy in the amplitude of power traces. The variance in the power traces from the FPGA is smaller than the power traces from Arduino UNO, which means that it requires a higher accuracy towards the amplitude of the power trace. The ADC on RASCv2 only has 8 bits, which limits its successful cracking rate and detecting traces from the FPGA.

5.3 Summary and Discussion

This experiment in Arduino UNO cracking shows the correctness and stability of the EM and power traces collected by RASC. The experiment in FPGA cracking shows that RASCv2 not only has the ability to crack microcontrollers such as Arduino UNO but also can crack FPGAs. The latter experiment better presents the value of RASC since FPGAs are more widely used not only in this area but also in the industry.

For time complexity, different subkeys require different numbers of traces, that is, different amounts of time. In Arduino experiments, some subkeys, such as #6 and #10, can be extracted with a high success rate in a short period of time. For example, RASC needs to collect only 2,000 traces and the whole cracking process is completed within 2 minutes. However, more challenging subkeys, such as #11 and #15, need as many as 20,000 power traces for 100% success. Compared with the Arduino UNO, RASC needs to collect more power traces for attacking the FPGA.

In the end, the purpose of this experiment was to verify the correctness of the EM and power traces collected by RASC, which we achieved. In addition, this experiment is a good start for the follow-up experiments and a good example to show the attack capability of the RASC.

6 MALWARE DETECTION RESULTS AND DISCUSSION

In order to demonstrate the defense capabilities of RASC, we performed a set of malware detection experiments for 10 different benchmarks. For every benchmark, there are four different scenarios: (1) original (i.e., malware-free), (2) buffer overflow, (3) code injection, and (4) ROP attack. The target board in this experiment is Arduino UNO. In this section, we include the description of each benchmark, the payload of the three different kinds of malware attacks, how to trigger three different kinds of malware-infected codes, and the overall results. To analyze the results, we examine two benchmarks more closely: one in which RASC obtains a high detection rate in all four scenarios and one example that RASC had trouble with.

6.1 Benchmarks

Each benchmark includes many loops of the numerical function. These numerical functions are critical in real applications such as face recognition, self-driving cars, mechanical manufacturing, and more. Any minor errors in these numerical functions could cause severe damage to property and loss of lives. The benchmarks that we use are as follows:

(1) **Circuit.** The benchmark "Circuit" loads the matrix of the testing circuit, which stands for the node and components in the circuit. TIt has an internal function to calculate the shortest path between two components. In the end, the results are presented with the function `printf`.

(2) **Cubic.** The benchmark "Cubic" calculates the distances between two different objects using a cubic function thousands of times and prints the results with the function `printf`.
(3) **Decode.** The benchmark "Decode" loads an input matrix and uses a predefined algorithm to decode the ciphertext. After decrypting the ciphertext, the plaintext will be output with the function `printf`.
(4) **Prime.** The benchmark "Prime" finds a prime number between a given range using the Wheel Seive method. After finding a large set of the prime number of a dynamic range, the results will be printed with the function `printf`.
(5) **Gausseidel.** The benchmark "Gausseidel" is a C++ program to implement the Gauss-Seidel method. In numerical linear algebra, the Gauss-Seidel method, also known as the Liebmann method or the method of successive displacement, is an iterative method used to solve a linear system of equations.
(6) **Gaussjordan.** The benchmark "Gaussjordan" is a C++ program to implement the Gauss-Jordan Elimination algorithm. In linear algebra, Gaussian elimination (also known as *row reduction*) is an algorithm for solving systems of linear equations.
(7) **Matrixdistance.** The benchmark "Matrixdistance" calculates the distances between two different matrices.
(8) **Matrixmultiply.** The benchmark "Matrixmultiply" multiplies two different matrices and print them out.
(9) **Signednumber.** The benchmark "Signednumber" multiplies two signed numbers using Booth's algorithm. Booth's algorithm is a multiplication algorithm for two signed binary numbers in two's complement notation.
(10) **Root.** The benchmark "Root" solves one quadratic equation with different inputs and prints them.

6.2 Payload and the Attack Model

There are three different malware attack scenarios included in this section: buffer overflow attack, code injection attack, and ROP attack.

For implementing the malware code, we assume that the attacker can access the running code and the ROM address of the device. For the buffer overflow attack, we assume that the attacker could call unprotected functions to overwrite important code parameters. For the code injection attack, we assume that the attacker could access the running code and insert malicious code. For the ROP attack, we assume that the attacker could load the malicious code to an unused part of the memory and call the malicious code in the main function.

The payload of the buffer overflow attack and the code injection attack in these 10 benchmarks aims to modify the frequency and time of the periodic loop to redirect the control flow of the target device or trap the target device into an endless loop. The ROP attack aims to call the preloaded malicious code to leak the critical data stored in the memory.

6.3 Trigger

(1) **Trigger for buffer overflow attack.** The buffer overflow attacks need to focus on two adjacent memory address spaces. For achieving this, we define two variables in one struct class at the beginning of the whole program. One variable is 'string' and another is 'int.' When we want to trigger the buffer overflow, the function `strcpy` is adopted to copy a long string to the short-sized string variable to modify the int variable to a large number.
(2) **Trigger for code injection attack.** For triggering the code injection attack, since we assume that the attacker could access the code, a few lines are inserted into the benchmarks to alter the value of some parameters or print some sensitive data.

Table 2. Experimental Results for Malware Detection on 10 Benchmarks in 4 Scenarios

List	Malware-free	Buffer overflow	Code injection	ROP
Cubic	1	0.06	0.06	1
Circuit	0.98	1	1	1
Decode	1	1	1	1
Gausseidel	1	1	1	0.84
Root	0.98	1	0.04	1
Prime	1	1	1	1
Gauss	0.98	0.06	1	1
Matrixdistance	1	0.02	0.52	1
Multiply	1	0.02	0.02	1
Signednumber	1	1	1	1

(3) **Trigger for ROP attack.** In the previous section, we assume that the attacker has preloaded the malicious code into an unused part of the memory. Thus, we preload a few malicious asm codes in the function `void setup()` since this function runs only once. When triggering the ROP attack, we simply insert one asm line to call the malicious code.

6.4 Experimental Results

The experimental setup and workflow is the same as the AES cracking experiment described earlier. The target board is the Arduino UNO. Details on how we do the training and testing can be found in Section 4.2.2.

For malware detection, we assume that the RASC is arranged near the target chip and attached to the power source of the target chip. Unlike the AES-subkey extraction experiment, no trigger signal is allowed in the benchmarks since the external monitor, like RASC, should not disturb the normal operation of the target device, which means that the power traces collected by RASC can start from any time point of the runtime. This increases the challenge of detecting malware codes, but it is closer to the real-world scenario.

The experiment results are presented in Table 2, which show a high detection rate for most of the benchmarks. However, a few failed. The spectral profiling-based SVM classifier has a very high success rate on ROP attacks since the ROP hijacks the control flow to a completely different code stored in the memory. In the benchmarks "Cubic," "Root," "Gauss," and "Matrixdistance," the spectral profiling-based SVM classifier sometimes cannot detect buffer overflow and code injection attacks since these attacks do not significantly alter the loop time and frequency of the periodic loop in the spectrogram. More illustrative discussions are provided in the next sections.

In conclusion, the trained SVM classifier is proven effective for most of the benchmarks and their related attacks. However, our method seemingly has two limitations. First, the accuracy of the spectrogram is essential to the success rate. The accuracy here is limited by the sensitivity of the 8bit ADC on RASCv2. The spectral profiling method is based on the amplitude-modulated principle; thus, the sensitivity can directly decide the correctness of the frequency and time of the loop. Second, the spectral method can be used only when the main code includes some periodic loops. The technique limits the range of doing malware experiments with RASC. This motivates future work in RASC that is more fine-grained, for example, side channel-based disassembly.

6.5 High Success Rate Example ("Gausseidel") and Discussion

The code of Gausseidel is presented in Figure 14(a). For the malware-free code, Gausseidel includes two periodic loops. The first loop runs 500 times for calculating the result of the first Gausseidel function with 500 loops. The second loop runs another 500 times for calculating the result of the

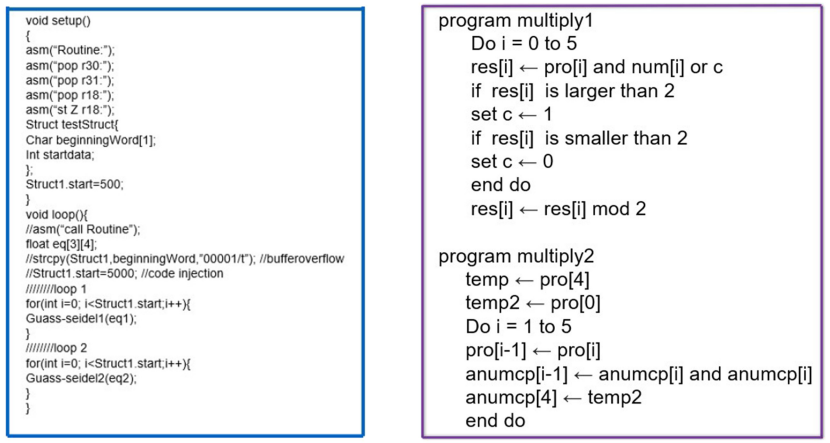

Fig. 14. Sample code for (a) benchmark "Gausseidel" and (b) benchmark "Multiply."

second Gausseidel function with 500 loops. The two Gausseidel functions have different loop times. The first Gausseidel function costs four clock cycles and the second Gausseidel function costs three clock cycles. For the buffer overflow attack, if we give the variable `Struct1.beginningWord` an oversized string, the value of the `Struct1.start` will be modified to a large number, which causes the wrong calculation of the benchmark. The code injection attack directly changes the value, and the result of the spectrogram is very close to the buffer overflow attack result. The ROP attack calls the preloaded asm code "Routine" and the whole program will run the malicious code only, never returning to the good code.

RASCv2 achieves a high successful recognition rate for malware-free code, buffer overflow attacks, ROPs, and code injection scenarios for Gausseidel. Malware-free code contains two different functions; thus, we should see two different loops with different frequencies. In Figure 15(a), two different frequency loops can be clearly seen in the spectrogram. However, the code injection attack and the buffer overflow attack increase the loop time of the first function. In this case, RASC detects only one loop. There are huge differences resulting from the ROP attack since the whole program is hijacked and the Arduino runs completely different codes.

6.6 Low Success Rate Example ("Multiply") and Discussion

Unlike Gausseidel, the benchmark Multiply has a very low detection rate on buffer overflow attacks and code injection attacks. A part of the code is presented in Figure 14(b). The programs multiply1 and multiply2 are two internal loops in the benchmark "Multiply." Each runs 300 times for a one-time run. The spectrograms for Multiply are shown in Figure 16. The reason why the classifier fails is because the frequency of the functions multiply1 and multiply2 are close to each other. The former function runs at 300 kHz and the latter function runs at 250 kHz. In the spectrogram of malware-free code, the first loop and the second loop are almost the same due to close frequency. The buffer overflow and malware attack in this benchmark modify the loop time of only two internal loops; thus, we can see only one loop in the spectrogram of malware code. The spectrogram of malware-free code and malware code are almost the same since the frequency of the two loops are nearly the same. In this case, the classifier cannot detect the buffer overflow attack and the code injection attack. For the ROP attack, however, the classifier can detect it because the ROP attack runs an entirely different program.

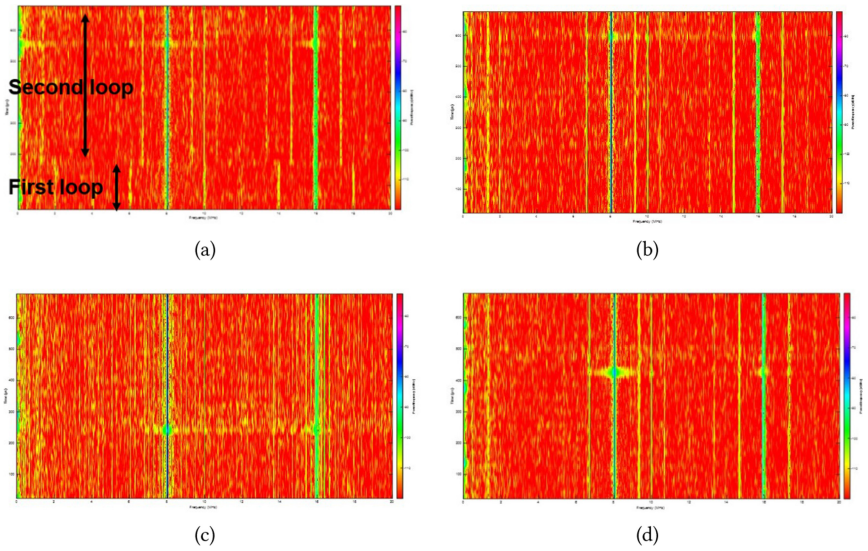

Fig. 15. Spectrograms for Gausseidel in (a) malware-free, (b) code injection, (c) ROP, and (d) buffer overflow scenarios.

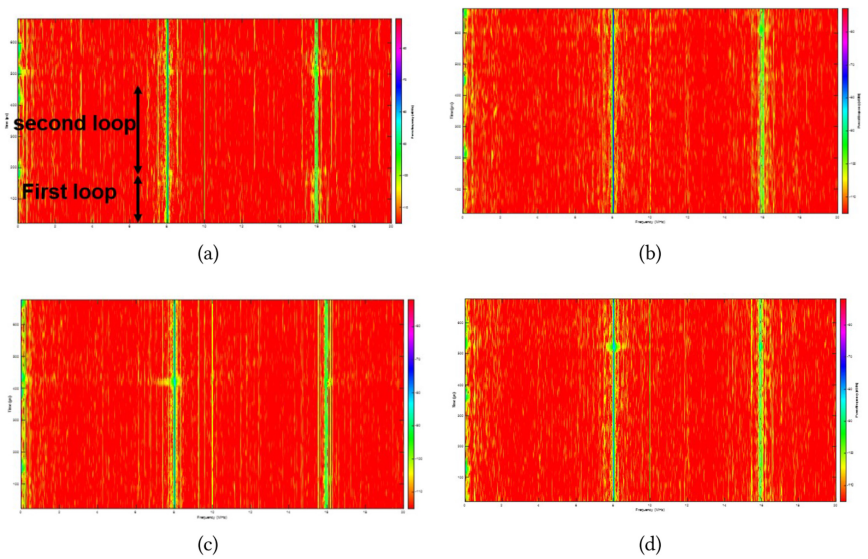

Fig. 16. Spectrograms for Multiply in (a) malware-free, (b) code injection, (c) ROP, and (d) buffer overflow scenarios.

7 CONCLUSION AND FUTURE WORK

In this paper, we demonstrated the attack and defense capability of RASCv2 with AES key extraction and malware detection experiments, respectively. Besides, RASCv2 shows its ability to transmit data remotely. All the experiments show the potential ability of RASC to perform remote side-channel monitoring. In future work, we plan to perform additional experiments and also

improve RASC's design in the third version based on what we've learned from v2. First, we plan to combine EM and power traces together. In RASCv2, two ADCs could run at the same time, which means it can gather EM and power traces simultaneously. We assume this characteristic could decrease the time to crack subkey since two different channels are used. Here should be noted the combination of two different channels is not a conclusion of this paper but an assumption that needs us to design experiments to verify it in the future. However, as experiment results show, the power traces break subkeys with fewer traces than EM traces. This caused us issues when we tried to combine EM and power traces together in v2. In this case, the new version of the EM antenna will be designed to balance the ability of cracking subkeys between EM and power channel. Second, we hope to move most of the processing used for malware detection from the PC/admin to RASC. For example, an internal FFT core will be implemented inside the RASC. RASC could do the data streaming and FFT to the power/EM traces at the same time. Moreover, algorithms will be inserted into RASC to increase the SNR of the EM and power signals. Third, we will also make updates to RASC's performance to handle instruction disassembly. In RASCv2, the low sampling speed limits the choice of the method used in the malware detection experiment. Moreover, the disassembly experiment has a higher requirement of the sampling speed since most of the methods adopted are fine-grained methods. In addition, the sensitivity decides the accuracy of the traces. This is an essential parameter for better malware detection results and the disassembly experiment in the next step. Finally, the wrong detection range of the EM ADC caused RASC to miss half of the range of the trace. An improved circuit will be implemented for RASC to measure negative voltages.

REFERENCES

[1] [n.d.]. *SAKURA-G*. Retrieved April 15, 2022 from http://satoh.cs.uec.ac.jp/SAKURA/hardware/SAKURA-G.html.

[2] 2008. *GPcode.AK Ransomware*. Retrieved April 15, 2022 from https://www.knowbe4.com/gpcodeak-ransomware.

[3] 2014. *Android/Ransom.Koler*. Retrieved April 15, 2022 from https://blog.malwarebytes.com/detections/android-ransom-koler/.

[4] 2014. *CTB-Locker (Curve-Tor-Bitcoin Locker) or Critoni.A*. Retrieved April 15, 2022 from https://www.knowbe4.com/curve-tor-bitcoin-locker.

[5] 2015. *Lockerpin Ransomware steals PINs, locks Android devices permenantly*. Retrieved April 15, 2022 from https://www.zdnet.com/article/lockerpin-ransomware-steals-pins-locks-android-devices-Permanently/.

[6] Manaar Alam, Sarani Bhattacharya, Debdeep Mukhopadhyay, and Sourangshu Bhattacharya. 2017. Performance counters to rescue: A machine learning based safeguard against micro-architectural side-channel-attacks. Cryptology ePrint Archive, Report 2017/564. Retrieved April 15, 2022 from https://eprint.iacr.org/2017/564.

[7] N. Nalla Anandakumar and S. Dillibabu. 2012. Correlation power analysis attack of AES on FPGA using customized communication protocol. In *Proceedings of the 2nd International Conference on Computational Science, Engineering and Information Technology*. 683–688.

[8] Fred Bazzoli. 2020. Digital thermometer data may provide insight into COVID-19 surges. Healthcare IT News. Retrieved April 15, 2022 from https://www.healthcareitnews.com/news/digital-thermometer-data-may-provide-insight-covid-19-surges.

[9] G. Becker, J. Cooper, E. De Mulder, G. Goodwill, J. Jaffe, G. Kenworthy, et al. 2013. Test vector leakage assessment (TVLA) derived test requirements (DTR) with AES. In *International Cryptographic Module Conference*.

[10] Noura Benhadjyoussef, Mohsen Machhout, and Rached Tourki. 2011. Optimized power trace numbers in CPA attacks. In *8th International Multi-Conference on Systems, Signals Devices*. 1–5.

[11] Robert Bridges, Jarilyn Hernández Jiménez, Jeffrey Nichols, Katerina Goseva-Popstojanova, and Stacy Prowell. 2018. Towards malware detection via CPU power consumption: Data collection design and analytics. In *17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE'18)*. 1680–1684.

[12] C.-C. Chang and C.-J. Lin. 2011. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2, 3 (2011), 1–27.

[13] Nick Christoulakis, George Christou, Elias Athanasopoulos, and Sotiris Ioannidis. 2016. HCFI: Hardware-enforced control-flow integrity. In *Proceedings of the 6th ACM Conference on Data and Application Security and Privacy* (New Orleans, Louisiana) *(CODASPY'16)*. ACM, New York, NY, 38–49.

[14] Shane S. Clark, Benjamin Ransford, Amir Rahmati, Shane Guineau, Jacob Sorber, Wenyuan Xu, and Kevin Fu. 2013. WattsUpDoc: Power side channels to nonintrusively discover untargeted malware on embedded medical devices. In *2013 USENIX Workshop on Health Information Technologies (HealthTech'13)*. USENIX Association, Washington, D.C. https://www.usenix.org/conference/healthtech13/workshop-program/presentation/clark.

[15] C. Cowan, C. Pu, D. Maier, J. Walpole, P. Bakke, S. Beattie, A. Grier, P. Wagle, Q. Zhang, and H. Hinton. 1998. Stackguard: automatic adaptive detection and prevention of buffer-overflow attacks. In *USENIX security symposium*, vol. 98. San Antonio, TX, 63–78.

[16] Ruan de Clercq and Ingrid Verbauwhede. 2017. A survey of hardware-based control flow integrity (CFI). *arXiv preprint arXiv:1706.07257* (6 2017).

[17] Philippe De Wilde. 2013. *Neural Network Models: Theory and Projects*. Springer Science & Business Media.

[18] Admir Dizdar. 2021. *SQL injection attack: Real life attacks and code examples*. Retrieved April 15, 2022 from https://www.neuralegion.com/blog/sql-injection-attack/.

[19] Thomas Eisenbarth, Christof Paar, and Björn Weghenkel. 2010. *Building a Side Channel Based Disassembler*. Springer-Verlag, Berlin, 78–99.

[20] Enes Göktas, Elias Athanasopoulos, Herbert Bos, and Georgios Portokalidis. 2014. Out of control: Overcoming control-flow integrity. In *2014 IEEE Symposium on Security and Privacy* (2014), 575–589.

[21] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown. 2001. MiBench: A free, commercially representative embedded benchmark suite. In *Proceedings of the 4th Annual IEEE International Workshop on Workload Characterization. WWC-4 (Cat. No.01EX538)*. 3–14.

[22] Yu Han, Xuecheng Zou, Liu Zhenglin, and Yi-cheng Chen. 2008. Efficient DPA attacks on AES hardware implementations. *International Journal of Communications, Network and System Sciences* 1 (1 2008), 68–73.

[23] Vincent Immler, Johannes Obermaier, Kuan Kuan Ng, Fei Xiang Ke, JinYu Lee, Yak Peng Lim, Wei Koon Oh, Keng Hoong Wee, and Georg Sigl. 2019. Secure physical enclosures from covers with tamper-resistance. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019 (2019), 51–96.

[24] I. T. Jolliffe and J. Cadima. 2016. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374, 2065 (2016), 20150202.

[25] Yu Jun Xiao, Wen Yuan Xu, Zhen Hua Jia, Zhuo Ran Ma, and Dong Lian Qi. 2017. NIPAD: A non-invasive power-based anomaly detection scheme for programmable logic controllers. In *Frontiers Inf Technol Electronic Eng*. 519–534.

[26] Vladimir Kiriansky, Derek Bruening, and Saman P. Amarasinghe. 2002. Secure execution via program shepherding. In *USENIX Security Symposium*.

[27] Peter A. Lachenbruch and Matthew Goldstein. 1979. Discriminant analysis. *Biometrics* (1979), 69–85.

[28] Mehari Msgna, Konstantinos Markantonakis, and Keith Mayes. 2014. Precise instruction-level side channel profiling of embedded processors. In *Proceedings of the 10th International Conference on Information Security Practice and Experience - Volume 8434* (Fuzhou, China) *(ISPEC'4)*. Springer-Verlag, Berlin, 129–143.

[29] Alireza Nazari, Nader Sehatbakhsh, Monjur Alam, Alenka Zajic, and Milos Prvulovic. 2017. EDDIE: EM-based detection of deviations in program execution. In *ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA'17)*. 333–346.

[30] Jungmin Park, Fahim Rahman, Apostol Vassilev, Domenic Forte, and Mark Tehranipoor. 2019. Leveraging side-channel information for disassembly and security. *J. Emerg. Technol. Comput. Syst.* 16, 1, Article 6 (Dec. 2019), 21 pages.

[31] Jungmin Park, Xiaolin Xu, Yier Jin, Domenic Forte, and Mark Tehranipoor. 2018. Power-based side-channel instruction-level disassembler. In *55th ACM/ESDA/IEEE Design Automation Conference (DAC'18)*. 1–6.

[32] Ronny Richardson and Max M. North. 2017. Ransomware: Evolution, mitigation and prevention. *International Management Review* 13, 1 (2017), 10.

[33] Nader Sehatbakhsh, Monjur Alam, Alireza Nazari, Alenka Zajic, and Milos Prvulovic. 2018. Syndrome: Spectral analysis for anomaly detection on medical IoT and embedded devices. In *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST'18)*. 1–8.

[34] N. Sehatbakhsh, A. Nazari, A. Zajic, and M. Prvulovic. 2016. Spectral profiling: Observer-effect-free profiling by monitoring EM emanations. In *49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'16)*. 1–11.

[35] N. Sehatbakhsh, A. Nazari, A. Zajic, and M. Prvulovic. 2016. Spectral profiling: Observer-effect-free profiling by monitoring EM emanations. In *49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'16)*. 1–11.

[36] Hovav Shacham. 2007. The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86). In *Proceedings of CCS 2007*, Sabrina De Capitani di Vimercati and Paul Syverson (Eds.). ACM Press, 552–561.

[37] H. Shen, M. T. Rahman, N. Asadizanjani, M. Tehranipoor, and S. Bhunia. 2018. Coating-based pcb protection against tampering, snooping, em attack, and x-ray reverse engineering. In *ISTFA*. ASM International, 290–294.

[38] Andrew Stern, Kun Yang, Jason Vosatka, Adam Duncan, Jungmin Park, Domenic Forte, and Mark Tehranipoor. 2019. RASC: Enabling remote access to side-channels for mission critical systems. In *GOMACTech*.

[39] Daehyun Strobel, Florian Bache, David Oswald, Falk Schellenberg, and Christof Paar. 2015. SCANDALee: A side-ChANnel-based DisAssembLer using local electromagnetic emanations. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE'15)*. 139–144.

[40] Daehyun Strobel, Florian Bache, David Oswald, Falk Schellenberg, and Christof Paar. 2015. Scandalee: A side-channel-based disassembler using local electromagnetic emanations. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition* (Grenoble, France) *(DATE'15)*. EDA Consortium, San Jose, CA, 139–144.

[41] NewAE Technology. [n.d.]. *ChipWhisperer-Lite XMEGA*. Retrieved April 15, 2022 from https://www.newae.com/chipwhisperer.

[42] Mohammad Tehranipoor and Farinaz Koushanfar. 2010. A survey of hardware Trojan taxonomy and detection. *IEEE Design Test of Computers* 27, 1 (2010), 10–25.

[43] Tektronix. [n.d.]. *Tektronix MDO3102 mixed domain oscilloscope*. Retrieved April 15, 2022 from https://www.tek.com/datasheet/mixed-domain-oscilloscopes.

[44] Texas Instruments. 2013. *ADC08200 8-Bit, 20 Msps to 200 Msps, low power A/D converter with internal Sampleand-Hold*. Texas Instruments.

[45] Muhammad Umair, Muhammad Aamir Cheema, Omer Cheema, Huan Li, and Hua Lu. 2021. Impact of COVID-19 on adoption of IoT in different sectors. *CoRR* abs/2101.07196 (2021). https://arxiv.org/abs/2101.07196

[46] Wikipedia. 2001. *Code Red (Computer Worm)*. Retrieved April 15, 2022 from https://en.wikipedia.org/wiki/Code_Red_(computer_worm)#cite_note-caida-3.

[47] Xilinx. 2018. *Spartan-3E FPGA Family Data Sheet*. Xilinx.

[48] Kun Yang, Domenic Forte, and Mark Tehranipoor. 2018. ReSC: An RFID-enabled solution for defending IoT supply chain. *ACM Trans. Des. Autom. Electron. Syst.* 23, 3, Article 29 (Feb. 2018), 27 pages.

[49] Ibrar Yaqoob, Ejaz Ahmed, Muhammad Habib ur Rehman, Abdelmuttlib Ibrahim Abdalla Ahmed, Mohammed Ali Al-garadi, Muhammad Imran, and Mohsen Guizani. 2017. The rise of ransomware and emerging security challenges in the Internet of Things. *Computer Networks* 129 (2017), 444–458.

[50] Man-Ki Yoon, Sibin Mohan, Jaesik Choi, and Lui Sha. 2015. Memory heat map: Anomaly detection in real-time embedded systems using memory behavior. In *Proceedings of the 52nd Annual Design Automation Conference* (San Francisco, California) *(DAC'15)*. ACM, New York, NY, Article 35, 6 pages.

[51] C. Zhang, T. Wei, Z. Chen, L. Duan, L. Szekeres, S. McCamant, D. Song, and W. Zou. 2013. Practical control flow integrity and randomization for binary executables. In *2013 IEEE Symposium on Security and Privacy*. 559–573.

Received July 2021; revised November 2021; accepted March 2022