Machine-Learning-based Side-Channel Attack Detection for FPGA SoCs

Lars Bauer, Hassan Nassar*, Nadir Khan, Jürgen Becker*, and Jörg Henkel*
*Karlsruhe Institute of Technology, Germany

***Abstract*—Embedded systems are threatened by side-channel attacks that allow the extraction of private keys from tampered systems. This particularly applies to FPGA-based SoCs that are widely used due to their attractive features like flexibility, etc. Power analysis (PA) attacks use power fluctuations that occur during cryptographic operations to incrementally reconstruct bits of a private key. Similarly, electromagnetic analysis (EMA) attacks use electromagnetic radiation. Countermeasures against PA and EMA attacks come at noticeable power, performance, and/or area overhead, and thus they should only be enabled when needed.**

This paper proposes a novel concept that uses machine learning (ML) to detect whether or not a system was tampered for a PA or an EMA attack. The main challenge is to distinguish a system that has been indeed tampered from a system that is untampered but operating under different conditions (e.g., higher ambient temperature) that may make it appear as if the system was tampered (e.g., removed heat sink), especially as we cannot trust off-chip sensors as they are under the control of the potential attacker (e.g., the end-user of the embedded system). Therefore, we can only use trustable information that can be acquired from within the system and we have to implement the entire measurement and classification flow within the FPGA-based SoC. We investigate, train, and deploy a lightweight on-chip ML-based approach along with an on-chip measurement infrastructure that uses load generators and the available on-chip sensors to distinguish tampered systems from untampered ones. For detecting EMA (PA) attacks, we reach a high accuracy, i.e., the number of correctly classified systems relative to all systems, of 0.9880 (0.9090) and a high precision, i.e., the number of systems correctly classified as tampered relative to all systems classified as tampered, of 0.9883 (0.9090). This comes at a reasonable resource overhead of $<1\,\%$ ($\sim 6\,\%$) of the available LUTs and an on-chip classification time of only $\sim 122\,\mathrm{ms}$ ($\sim 40\,\mu\mathrm{s}$). The entire system including the on-chip measurement infrastructure, the ML-based classification, and the tampering for PA and EMA attacks are implemented and evaluated on FPGA boards.

***Index Terms*—Side Channel Detection, Machine Learning, Hardware Security, FPGA, PA, EMA.**

I. Introduction

Smart technologies like IoT, and Industry 4.0 are emerging industrial fields that need cheap, secure and flexible systems. [1], [2]. FPGA-based SoCs (e.g., Xilinx' Zynq 7000 or UltraScale+ MPSoC families) provide flexibility and high performance to fulfill these demands [2]. Control algorithms, user interfaces, etc. can be implemented in software on the CPUs, whereas the reconfigurable logic of the FPGA can be used to implement hardware accelerators for the computationally-intensive parts of the applications [3]. In general, FPGA-based SoCs are widely used due to their attractive features like flexibility, etc. [2], [3]. They are also used in security-relevant environments [4], for instance, IoT sensor data needs to be encrypted before it can be transmitted, devices need to be authenticated after they are deployed, etc.

Embedded systems are threatened by *side channel cryptanalysis* (SCC). Cryptographic operations (e.g., AES en-/decryption) lead to observable power fluctuation and electromagnetic radiation that can be used to extract information via unintended *side channels* [5], [6]. Therefore, *power analysis* (PA) attacks and *electromagnetic analysis* (EMA) attacks are the most prominent SCCs in this area (see Section II). For PA, a system is tampered by placing a shunt resistor on the power line to measure power consumption variations [7], [8]. For EMA, a system is tampered by removing the fan, heat sink, and heat spreader in order to place field probes close to the chip to measure its electromagnetic radiations [9], [10]. Eventually, SCCs allow the extraction of private keys from tampered embedded systems such as FPGA-based SoCs [9], [11] (more background is provided in Section II).

Several countermeasures exist against SCCs, but they all incur a noticeable power, performance, and/or area overhead [12], [13]. Therefore, **this paper aims at detecting PA and EMA attacks** to enable countermeasures only when needed. But most existing side-channel detection techniques require modifications with special sensors or tampering resistance [5], [14]. That means that they cannot be applied to commercial embedded systems that use commercial-off-the-shelf (COTS) FPGA-based SoCs, because they do not provide such tamper resistance (except some military-grade FPGAs [15], [16]) and we cannot trust off-chip sensors, because they are under the control of the potential attacker (e.g., the end-user of the embedded system).

Our **main idea to be able to detect PA and EMA attacks** is to identify manipulations that are needed to enable SCCs (even though we cannot tell whether or not an SCC is ongoing). We aim at detecting tampered systems (i.e., a shunt in the power supply for PA attacks or a removed fan, heat sink, and heat spreader for EMA attacks) by only utilizing the available SoC-internal sensors for supply voltage and temperature, but without making any assumptions about the ambient environment, the FPGA power supply, the PCB design, etc.

When an attacker removes the fan, heat sink, and heat spreader to prepare for an EMA attack, then that reduces the thermal capacity of the SoC. That means that we will observe

This work was partially funded by the Federal Ministry of Education and Research, BMBF, as part of the MANNHEIM-CeCaS project (grant: 16ME0817) and the DI-EDAI project (grant: 16ME0990K).

a generally higher temperature and also a steeper temperature rise upon enabling a load (e.g., performing some on-chip calculations). Similarly, with a shunt in the power supply, we will observe a larger load-dependent voltage drop. While these measurable observations are indications of a tampered system, **our main challenge is to distinguish manipulation from normal scenarios**. That means that we need to distinguish whether an observed higher temperature is due to a PA manipulation or due to higher ambient temperature, a slower fan etc. However, we cannot reliably know the ambient temperature and fan speed, because all off-chip sensors are under the control of the potential attacker.

Along these lines, **our main contributions in this paper are:**

- A **novel concept** that uses side effects of SCC-enabling manipulations to identify whether or not a system is tampered, by only using trustable information that can be acquired from within the system itself.
- Investigating, training, and deploying a **light-weight on-chip machine learning (ML) approach** to successfully distinguish PA and EMA manipulations for tampered systems from untampered systems in various ambient scenarios (that are unknown to the ML approach).
- Developing an **on-chip measurement infrastructure** that employs only typical on-chip sensors of commercial FPGA-based SoCs.

The paper is organized as follows: Section II presents the necessary background for PA/EMA attacks and the attacker model, and Section III discusses related approaches for PA/EMA detection and their limitations. Section IV details our main contributions for the monitoring infrastructure and the machine learning approach. Section V evaluates the effectiveness of our approach and its overheads, and Section VI draws conclusions.

II. Background and Attacker Model

Physical attacks gained more attention in recent years as they do not rely on the existence of cryptographic imperfections, but can also be used to attack computationally secure systems. Physical attacks observe physical quantities to extract information such as keys. They are especially effective against embedded systems, as they expose their hardware to adversaries, which provides full physical access and relaxed timing constraints for performing the attack.

Side-channel attacks are classified as passive non-invasive physical attacks [17]. They monitor one or multiple physical parameters such as power, electromagnetic (EM) radiation, temperature, or timing [18]–[21]. 'Passive' means that the system is operating under normal conditions within its specifications, e.g., no especially prepared input data to trigger corner cases, no injected glitches in the power supply or clock signal etc. 'Non-invasive' means that the semiconductor chip is not manipulated, i.e., no secret data from ROM or embedded flash memory can be accessed and no on-chip signals can be probed or manipulated. Removing the heat sink, adding a precision shunt in the power supply, or other manipulations that do not demand expensive semiconductor equipment are not considered as invasive attacks but are sometimes called semi-invasive.

When a charge is applied to or removed from a CMOS transistor gate, electrons pass the silicon substrate, consume power, and emit EM radiation. The switching activity (and thus the power consumption and EM radiation) of CMOS circuits is highly data-dependent. This dependency can be exploited in order to retrieve information about secrets such as keys. The power analysis (PA) side-channel attack uses power fluctuations that occur during cryptographic operations to incrementally reconstruct the bits of a private key. With rising clock frequencies, the power consumption needs to be measured with high data rates and accordingly high-frequency bandwidth (i.e., the bandwidth of a signal in the frequency domain). With more recent technology nodes, the dynamic power consumption of CMOS logic reduces, and thus the setup needs to be able to measure signals with a small amplitude in a noisy background. While the power consumption of a chip can be measured in many different ways, the above-mentioned constraints reduce the available options significantly. For PA attacks, typically a shunt (i.e., a resistor for high power with a small and precisely-known resistance) is added in series with the chip. The current consumed by the chip also flows through the shunt and the corresponding voltage drop across the shunt can be measured by a middle/upper-class oscilloscope.

There are different methods to implement PA attacks, e.g., simple power analysis (SPA), differential power analysis (DPA), or correlation power analysis (CPA) [22]. The SPA directly interprets the measured power consumption and is suitable when only a small number of power measurements is available, but it is unsuitable in case of noisy measurements. The DPA uses a hypothetical statistical model of the target device to predict its power consumption. It uses known plain text attacks to extract the cryptographic secret. The CPA is a variation of DPA and uses three phases, i.e., prediction, measurement, and correlation. It then calculates the correlation between predicted and measured power consumption. A high correlation indicates that the key hypothesis (used for the prediction) was correct.

The EM analysis (EMA) side-channel attack is based on Maxwell's equations and the relationship between moving charges and magnetic fields [19]. Similar to measuring power for PA, the EM field is measured for EMA. As the EM field also depends on the switching activity of CMOS circuits (and thus the private secret in case of cryptographic operations), EMA attacks can be as successful as PA attacks, or even more successful. PA suffers from a reduced signal-to-noise ratio in more recent technology nodes due to reduced core voltage and on-chip decoupling capacitors, and therefore EMA has become the greater threat for these devices [23]. For some special cases, far-field probes without physical access can be used for EMA attacks [9], [11]. But especially for complex designs (e.g., with multiple CPUs, IP-cores, embedded FPGAs etc.) near-field probes provide the significant advantage of adding spatial dimensions to the measurement (i.e., the surface area of the chip). That allows to focus the measurement on the area of interest, but requires fan, heat sink, and heat spreader to be removed [19], [24]. Similar to PA, different EMA variations

like simple EMA or differential EMA exist [25].

A. Attacker Model

We target embedded systems that use FPGA-based SoCs and that are deployed in the field, e.g., in the (industrial) IoT domain such as in smart meters, smart home automation, smart city infrastructure, wearable devices, communication devices, etc. We assume that we can trust the developers of the embedded system and the developers, manufacturers, and suppliers of all its components and used tools. We furthermore assume that the attacker has acquired the system, has full physical access to it, and aims to extract a secret key from it. We exclude unprivileged user applications accessing the system monitor. The on-chip sensors have a maximum frequency of 5.2 MHz [26] and lower resolution than an oscilloscope. Additionally, repeated reads of on-chip sensors can be halted by incremental delays, similar to memory contention on spinning processors [27], when the system monitor is accessed at abnormal frequencies. To clarify, there are two different kinds of secret keys involved: The *configuration key* is a secret key used to configure/boot the FPGA-based SoC and the *design key* is a secret key that is part of the application-specific design generated by the embedded system developer. In the following, we explain why we assume the configuration key to be safe and focus on the design key instead.

All recent generations of FPGA-based SoCs provide a secure boot mechanism via a configuration key, and we assume that the developer uses it. For instance, ref. [28] describes the available protection features for Xilinx' UltraScale and UltraScale+ families (but older FPGA families and FPGA families from other vendors provide similar features). The secure boot mechanism configures a configuration key inside the SoC and then generates authenticated and encrypted system files like the bitstream to configure the FPGA, boot loaders for the software, etc. It also ensures that the configuration of the SoC cannot be read back after it was configured. The secure boot mechanism uses a dedicated SoC-internal decryption engine that may even provide hardware resistance against different types of physical attacks [28]. So we assume that we can trust the secure boot mechanism, i.e., the configuration key and the entire configuration process.

When the system developer aims at performing encryptions/decryptions as part of the functionality of his embedded system, then he cannot use the dedicated SoC-internal decryption engine for that purpose. Even if the SoC-internal decryption engine would be available for purposes other than configuring the FPGA, it only performs decryption, but not encryption. Instead, the system developer has to implement his own en-/decryption in hardware and/or software and store a corresponding secret design key somewhere, for instance in the encrypted bitstream of the SoC. As the secret design key cannot be extracted by an attacker from the encrypted bitstream, it is only available inside the FPGA-based SoC after it was configured with the encrypted bitstream, and it is only accessible by the hardware-/software design of the SoC developer. The attacker may potentially upload a custom bitstream to the FPGA-based SoC and thus will gain control over it. However, this custom bitstream does not contain the secret design key (which is inside the encrypted bitstream) and so it does not help in gaining access to it. As our proposed ML-based side-channel attack (SCA) detection method also resides in the encrypted bitstream, it will always be available, when the secret design key is usable, i.e., when the FPGA-based SoC is configured with the encrypted bitstream.

In summary, we target scenarios, where the attacker has no simpler way to extract the secret design key other than using side-channels attacks. We assume that the attacker is budget constrained and thus can only apply non-invasive or semi-invasive attacks, i.e., PA and EMA side-channel attacks are the most promising candidates. And as the attacker has full physical access to the system, we assume that the attacker has all means to perform these attacks, i.e., he can tamper the system to measure power and temperature, he can trigger the system to perform encryption operations (to have something to measure), and he can perform known-plaintext attacks (needed by some SCAs), e.g., by manipulating sensor values that the system reads, encrypts and intends to communicate to a server.

III. Related Work

Several works already discuss side-channel attacks (SCAs), their types, how to detect them, and their countermeasures. In addition to EMA and PA, also other SCAs exist. Examples are cache-based SCAs [29], [30] and optical SCAs [31]. Additionally, the detection and countermeasures of these SCAs are also discussed in some works. For instance, [32] used machine learning (ML) to detect cache-based SCAs, [33] calls dummy functions to perturb the timing behavior and the cache access patterns, and [31] uses a novel memory design with insulator layers to prevent optical SCAs. These SCAs are out of the scope of this work. Cache-based SCAs target systems with a CPU and detect their hit and miss behavior. Optical SCAs target non-volatile memories, which are not part of the FPGA structure.

A. Anti-Tamper Mechanisms

As mentioned in Section II, EMA requires the fan, heat sink, and heat spreader to be removed, and PA involves adding a resistor to the voltage supply path. That is why one method to detect or counter both attacks is to have anti-tamper mechanisms implemented on the chip. This ranges from having anti-tamper monitors to tamper-resistant chips. Anti-tamper monitors are implemented based on sensors readings [34]. If a significant change in the temperature or voltage is detected and exceeds a certain threshold, an alarm is evoked. However, the detection is limited by the threshold, making it possible to perform attacks if they do not lead to significant voltage or temperature changes. In fact, the manipulations for EMA and PA attacks that are performed in this work are not detected by the anti-tamper monitors of the FPGA board used for testing.

The other form of anti-tamper mechanisms is tamper resistance [14], [35]. There, dismantling can be detected via sensors, reed switches, infrared signatures, X-ray detectors, or ultrasonic signatures. This is infeasible for normal COTS FPGAs as it results in having high power consumption and

increased production costs. Moreover, such techniques are not stable. Their performance degrades with time and can lead to the failure of the device itself [14].

Another resistance technique is using a barrier layer attached to the device to make it impossible to remove this layer without destroying the device. This barrier layer can be made of ceramic or steel [35]. The main drawback of this solution is that it has a high implementation and power cost [36]. Moreover, this solution is not easy to generalize for COTS devices. As mentioned in Section I, such costly solutions are only available for some military-grade FPGAs.

B. SC Countermeasures

Some works go a step further to implement countermeasures against SCAs. The countermeasures are active either during the whole runtime of the system, or only when security-critical applications are running [37], [38]. The countermeasures can be either lowering the Signal to Noise Ratio (SNR) [19], masking [39], or balancing with complementary logic [13]. Lowering the SNR can be achieved by different methods. The first method is to add additional noise to the signal. For CPUs this can be achieved by adding random timing shifts between instruction execution [40] in order to make the calculated mean powers not corresponding to the respective instructions. The addition of the random shifts is not trivial, as it needs to ensure that they are not easily reversible and they can affect the SNR significantly. Another approach is to use the Dynamic Voltage Frequency Scaling (DVFS) management system implemented on the chip to change the SNR [38]. However, this method is limited to systems that have on-chip DVFS and allow the security-critical applications to control it.

Masking is achieved by splitting an intermediate variable into multiple shares which are calculated with a random mask and are consequently harder to predict. Each share is required to be equiprobable distributed and all possible subsets of the shares need to be statistically independent of the intermediate variable. The computations are then performed on the shares without using the original intermediate variable [13]. Most masking techniques work only against SPA but fail against DPA or CPA, i.e., they only work against simple attacks [12], [39]. Moreover, to have masking work against higher-order attacks, the area overhead can reach $200\,\%$ of the original design as in [39].

Balancing is accomplished by adding dummy logic that executes (cryptographic) operations in parallel to the running cryptographic cores [41]. Thus balancing the static and dynamic power consumption of the system to effectively hide the power-related information, which makes it hard to detect it. Similar to masking, its protection is not fully ensured and they also have a high area overhead which reaches $646\,\%$ in [41].

Clearly, the countermeasures come with penalties either as time overhead (case of lowering the SNR) or as area overhead (case of balancing) or with both (case of masking). Therefore, it is beneficial to only enable them if an attack is detected. That's why using a detection method like the work presented here is crucial.

C. PA and EMA Detection

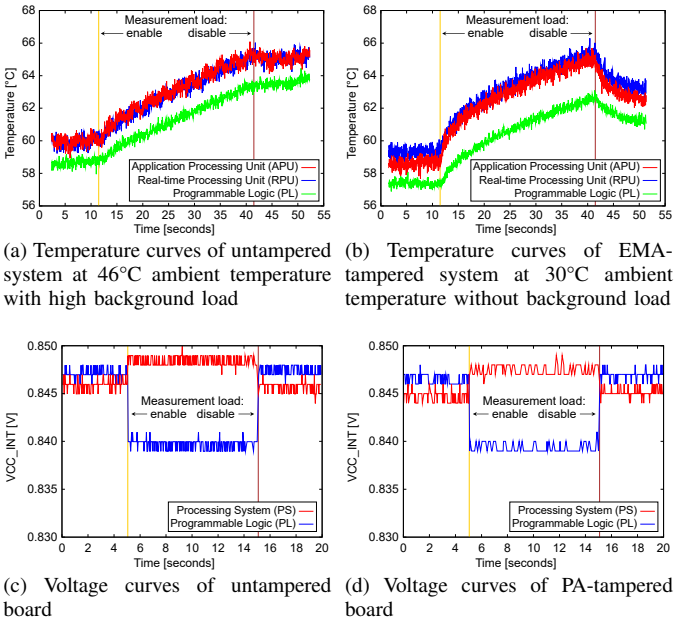
Other works have considered different PA and EMA detection techniques. However, most of these detection techniques can only detect either EMA or PA, in contrast to this work, which can detect both. For example, the technique proposed in [42] detects only EMA. It uses several monitors available for COTS computers like wireless activity monitors, temperature sensors, CPU load, and currently active communication channels. Based on the collective readings of these monitors, it can detect if an EMA is running against the computer or not. The main limitation of this technique is that it relies on the availability of many monitors that come with COTS computers. But it cannot be extended to other devices like FPGAs as they do not provide that many monitors.

Additionally, some of the detection techniques are limited by the need of adding custom sensors and circuitry on the chip. The EMA detection proposed in [36] needs the addition of special LC-based sensors to detect the probe's existence near the chip. As when the probe approaches the chip, the frequency of the LC sensors will change due to the mutual inductance. However, COTS systems typically do not provide such sensors.

The PA detection methods proposed in [5], [8] add multiple voltage sensors that are measuring the Power Distribution Network (PDN) and that need one exclusive analog-to-digital converter (ADC) per voltage sensor. They use the ADC-converted sensor readings as an input for an ML algorithm to detect whether or not a measurement shunt was added. Both works are similar and mainly differ in their sensor placement. Ref. [5] empirically places the sensors to cover a PDN with 10,000 nodes, while [8] uses an algorithm to decide the sensor placement on the PDN. However, COTS systems typically do not provide such sensors. Moreover, as both rely on simulation results only, they use ideal models for the voltage sensors and the ADC, which might affect the results when implemented on a real chip.

Another example is the approach proposed in [7] that detects PA. It uses ring oscillators (ROs) as a sensor. When a measurement shunt is added to the supply voltage path it has a voltage drop over it. Thus the voltage supplied to the ROs is lowered by this voltage drop. This change of the voltage level causes the frequency of the ROs to shift and this frequency shift is used as the sensor to detect the PA. However, the RO frequency can shift for other reasons, e.g., due to a changed temperature. This is especially problematic, as the ROs themselves lead to self-heating when they are active. This puts a limitation on the minimum resistance value of the measurement shunt that can be detected without facing a too high false-positive rate. In Section V, we compare against the RO-based [7] and ML-based [5], [8] PA detection techniques, as they have a similar goal to our techniques and use common metrics that can be compared.

The current state-of-the-art methods show the need for a more comprehensive solution, as the anti-tamper mechanisms are limited and most of them are unsuitable for COTS devices. Moreover, the SCC countermeasures have significant performance and area penalties. Therefore using them only when



(a) Temperature curves of untampered system at 46°C ambient temperature with high background load

(b) Temperature curves of EMA-tampered system at 30°C ambient temperature without background load

(c) Voltage curves of untampered board

(d) Voltage curves of PA-tampered board

Fig. 1: Temperature and voltage curves of tampered and untampered Xilinx ZCU104 boards

tampering is detected can avoid these unnecessary penalties. Finally, the existing methods for the detection of EMA and PA lack certain features that we provide. For instance, no method is capable of detecting both attack types by the same detection system. Additionally, most techniques require modification of the chip itself or resources that are not available in COTS FPGAs.

IV. Our proposed ML-based Side-Channel Attack Detection

A. Is ML even needed?

The hardware manipulations (see Section I) required to perform successful PA and EMA attacks (see Section II) change the voltage and temperature responses to load changes. For instance, when the fan, heat sink, and heat spreader are removed for an EMA attack, then the chip temperature will rise noticeably. So the question arises whether simply comparing the measured on-chip temperature against a predetermined threshold might already be enough to distinguish a tampered system from an untampered one. To investigate whether such a trivial classifier might already be sufficient for our requirements, we performed initial tests and measurements, based on the same experimental setup and hardware manipulations that are described for our evaluations in Section V-A.

Figure 1 shows the temperature (top row) and voltage (bottom row) measurements for an untampered (left column) and tampered (right column) system. We enable a *measurement load* for a few seconds (see vertical lines in Figure 1) and monitor the temperature and voltage response. If we would remove the fan, heat sink, and heat spreader under otherwise identical conditions, then the temperature difference would indeed be clearly visible. However, in Figure 1 we operate the tampered and untampered systems under different ambient temperatures and with different background loads (i.e., the CPU and/or FPGA activity inside the chip), which also influences the chip temperatures significantly. Note that our side-channel attack (SCA) detection system cannot reliably know the ambient temperature (as we cannot trust off-chip sensors as they are under the control of the potential attacker, see Section II-A) and the background load heavily depends on what the system is used for, what it is currently doing, what was the input data etc.

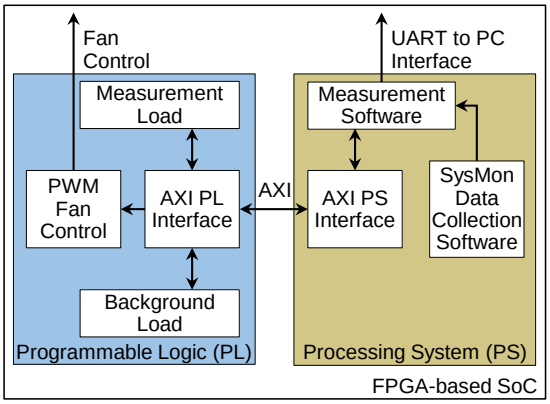
Figure 1 (a) and (b) show that we cannot use a simple threshold comparison against the measured temperature to decide whether or not the board was tampered. The temperatures at the beginning and the end of the measurements are very similar when comparing the untampered against the tampered system. The temperature change due to tampering is largely compensated by the different ambient temperature and background load. The shapes of the curves are partially differing in this example (e.g., directly after enabling the measurement load), however, there are also other scenarios (ambient temperature, fan speed etc.), where the shapes are more similar, but the absolute temperature values are less similar etc. Figure 1 (c) and (d) show how the supply voltage changes when activating the measurement load for an untampered and a tampered board, respectively. Here, the changes are even subtle under identical ambient conditions. Altogether, it is not clear how to manually implement a classifier to distinguish measurements from tampered and untampered systems, and therefore we are investigating ML-based methods in the following.

B. Our Hardware- and Software Architecture for Training and Detection

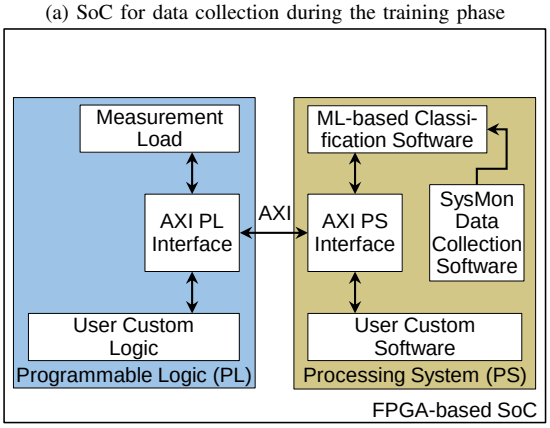
We need two different configurations of the FPGA-based SoC: one for data collection during the training phase and one for SCA detection in the finally deployed embedded system. They are sketched in Figure 2 and will be explained in the following. For the SCA detection, we must only rely on the available on-chip sensors, whose readings cannot be manipulated by a potential attacker. On-chip temperature sensors (to detect the thermal behavior) and voltage sensors (to detect the voltage drop behavior) are available on many FPGA-based SoCs. For instance, Xilinx' Zynq-7000 or UltraScale+ MPSoC families provide system monitoring (SysMon) capabilities that use internal sensors, internal voltage reference, and an internal analog-to-digital converter (ADC). They are accessible by software running on the CPUs in the processing system (PS) and can measure temperatures and voltages of different SoC components (for instance, the measurements shown in Figure 1). We implemented the configuration and measurement of temperature and supply voltage in the "SysMon Data Collection Software" module shown in Figure 2 (a) and (b).

In addition, we need a "Measurement Load" (see Figure 2) to induce stress and trigger a voltage/temperature response that we can measure. It is built of multiple ring oscillators (ROs) that can be enabled/disable by software. The actual values for all parameters (e.g., number of ROs) are given in the experimental setup (see Section V-A).

These are already all actors and sensors that are used for detection. But for training we need additional hardware- and software modules to set up and measure different scenarios. As shown in Section IV-A, different ambient temperatures



(a) SoC for data collection during the training phase



(b) SoC for detection during normal operation

Fig. 2: SoC configurations for (a) training and (b) detection

can mask the effects of a removed heat sink etc. Therefore, we need to collect measurement data under different ambient temperatures for training our ML models. The ambient temperature is controlled by placing the FPGA Board in an oven with configurable temperature. Whenever the desired ambient temperature changes from one measurement scenario to the next one, the measurement waits for a sufficiently long time to allow all SoC components to heat up/cool down to the new ambient temperature before the measurements are continued.

For systems without EMA tampering, there may be a fan mounted on the heat sink of the chip. We implemented a "PWM Fan Control" (see Figure 2 (a)) in hardware (configurable by software) to control the fan speed via an I/O pin and an off-chip driver circuit. Finally, we implemented a configurable "Background Load" generator (see Figure 2 (a)) to simulate different degrees of activity of the embedded system, as that will also influence the measured temperature and voltage. Everything inside the FPGA-based SoC is under the control of the developer of the embedded system (as explained in Section II-A) and the potential attacker can only control the ambient settings. Therefore, instead of learning the effects of different background loads indirectly, it would have been possible to query each hardware- and software module in the system to inform our ML-based SCA detection method about its current degree of activity. But that would have complicated the design of the embedded system significantly, as then the developer would have to provide all these interfaces between the modules of the embedded system and our ML-based SCA detection. Instead, we prefer to have as few interactions between our hardware- and software modules for attack detection and the actual system functionalities of the embedded system as possible, which also eases the deployment in already existing embedded systems significantly. We decided to treat the actual background load as unknown to our ML-based SCA detection method, i.e., it is not explicitly communicated, but only indirectly observable by the temperature and voltage measurements. For training, we used ROs (similar to those in the "Measurement Load") to implement the "Background Load". The "Measurement Software" (see Figure 2 (a)) can configure how many of them should be activated to simulate different degrees of background load.

The SoC configuration for data collection during training combines all the above mentioned components (see Figure 2 (a)). The "Measurement Software" configures the hardware for one scenario after the other, enables/disables the "Measurement Load", reads the measured values from the "SysMon Data Collection Software" and transfers the raw data to a PC via a UART connection. The actual training of the ML-based classifiers is done offline on a PC using the "scikit-learn" library in Python.

The SoC configuration for SCA detection (shown in Figure 2 (b)) removes all training-only components, e.g., the "PWM Fan Control" and the "Background Load". Instead, the "ML-based Classification Software" is now deployed on the processing system, i.e., the actual classification happens at runtime inside the FPGA-based SoC without relying on any off-chip component. Additionally, the "User Custom Logic" and the "User Custom Software" are now integrated, i.e., the parts that implement the functionality of the embedded system, which are not related to our ML-based SCA detection. In the following, we will discuss the different ML models that we considered and trained along with the features that we calculated from the sensor measurements and used as inputs for them.

C. Our ML-based Side-Channel Attack Detection

Our main step to detect side-channel attacks (SCAs) is to build the ML model. Our flow for building the model includes the tasks of feature extraction, scaling, selection, model training, validation, and evaluation. The measured data are split into a training dataset and test dataset before the feature extraction in order to avoid any bias towards the test dataset. The datasets contain various time series of several measurement scenarios from the different temperature and voltage sensors available on the COTS FPGA. The feature extraction, scaling, and selection are performed with the scikit-learn library, which can automatically extract hundreds of different features and filter them according to their relevance. In order to ensure that the scikit-learn library does not automatically perform feature extraction, scaling, and selection for the test dataset again (which would bias the results towards the test dataset), we explicitly made sure that the Extractor, Scaler, and Selector that were chosen for the training dataset are also used for the test dataset. After the features are extracted, multiple variations of selectors, scalers, and models are built and evaluated to retrieve the best performing combination.

For the feature selection, five selectors from the scikit-learn library are used, i.e., SelectKBest, SelectFpr, SelectFdr, SelectFwe, and SelectPercentile. The SelectKBest function takes k (the number of top features to select) as an input parameter and selects the features with the k highest scores. The SelectFpr, SelectFdr and SelectFwe functions are based on the false positive rate, false discovery rate, and family-wise error rate, respectively. They are configured by the input parameter $alpha$ (i.e., the significance level) and select the feature with the highest p-value (null hypothesis testing). The SelectPercentile function selects the features according to the percent of features to keep, which is set by the $percentile$ input parameter. By using these five selectors, we cover two typical methods for choosing the features. The first method is based on choosing the best performing feature (SelectKBest and SelectPercentile), and the second method is based on optimizations to reduce an error (SelectFpr, SelectFdr, and SelectFwe). The scikit-learn library provides a third method that is based on manually annotating the features with weights to express personal preference. For instance, high weights can be assigned to easily computable features or to features that are known to fit well to the targeted problem. As we had no explicit preference about which feature to use and as we did not want to interfere with the automatic feature selection, we decided not to use this option.

After the features are extracted and selected, an ML classifier has to be chosen. The classifiers considered in this work are the decision tree (DT), random forest (RF), extra-trees (ET), k-nearest neighbors (KNN), support vector machine (SVM), AdaBoost, and Gaussian naive Bayesian network (GNB). Two variations of the SVM are used: the C-Support Vector Classification (SVC), and the Number-controlled Support Vector Classification (NuSVC) that extends the SVC by a control parameter for the number of support vectors. We choose the considered classifiers based on two factors. The first factor is that they can be easily ported to embedded systems, which is done by either using the sklearn-porter for embedded models or by manually implementing them (which is what we did for KNN and GNB). The second and more important factor is to cover many types of supervised machine learning classifiers. SVM is covered by SVC and NuSVC, naive Bayesian by GNB, ensemble learning by AdaBoost, instance-based by KNN, and logic-based by DT, RF, and ET. Note that we aim for solutions that are lightweight enough to be used on the device itself with low latency overhead. And as the above-described classifiers perform very well for our use case (see Section V-B), we did not investigate Convolutional Neural Networks (CNNs).

After training and choosing the classifier, k-fold cross-validation (provided by scikit-learn) is used to avoid overfitting. The data is split into the amount of wanted folds while preserving the percentage of samples for each class. It takes the model, the training data set, and the folds as inputs. Accuracy is used as parameter to evaluate the overfitting. As a result, we obtain the classifier used to detect attacks (tampered vs. untampered), which is not limited to the training set.

D. Our Decision Tree Classifier for PA Attack Detection

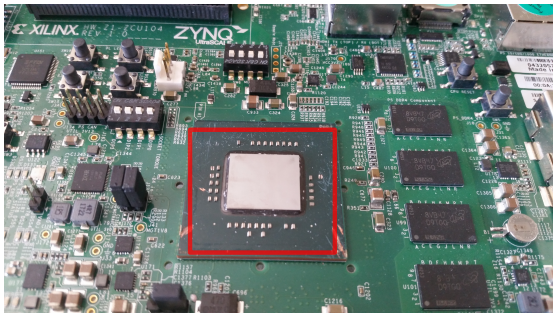
As will be shown in Section V, the decision tree (DT) classifier performed best and therefore it will be briefly explained here. DT classifiers are logic-based learning classifiers that use logic terms to represent the ML problem [43]. They are classifiers that can be easily understood and interpreted and that provide good performance with relatively small computational effort. DTs are typically built (i.e., trained) by using a divide and conquer approach, beginning with the most relevant features. To classify an input measurement, each node in the DT corresponds to a simple test that decides in which branch (starting in that node) to continue. The leaves of the tree terminate the search and are annotated by the classification decision. Other tree-based algorithms such as random forest (RF) or extra trees (ET) train multiple DTs and determine the classification decision by calculating the majority over the classification decisions of all individual trees. Trees in RF are built by splitting nodes using the best feature among a subset of randomly chosen features, whereas ET splits randomly.

V. Results

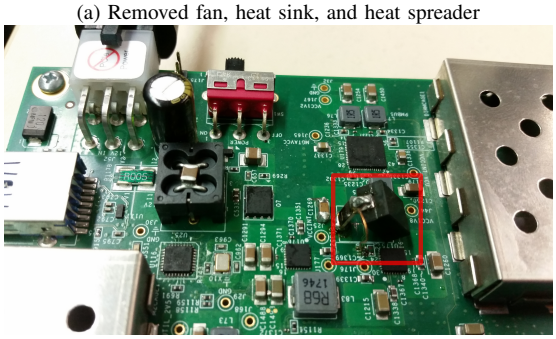
A. Experimental Setup

For our evaluation, we used two Xilinx ZCU104 boards with the Zynq UltraScale+ XCZU7EV MPSoC. We first measured both boards in their untampered form and then tampered one for PA attacks and the other for EMA attacks. For our measurements, we track all five sensors available on chip, the PL voltage, the PS voltage, the temperature of APU, the temperature of RPU, and the temperature of PL. Later when we apply feature selection, some of the sensors are excluded as they were not of statistical relevance. Figure 3 shows the tampered parts. For EMA tampering, we removed the fan, heat sink, and heat spreader of the chip (see Figure 3 (a)). And for PA tampering we inserted a $15\,\mathrm{m}\Omega$ precision shunt right after the power inductor of the board's boost converter for providing the VCC_INT power supply for the programmable logic (see Figure 3 (b)). As this was the only point we found on the board layout where the entire VCC_INT current had to pass by, we had to remove the power inductor and use its footprint on the board to mount both, the power inductor and our measurement shunt for the attack. As shown in the figure, one terminal of the power inductor and one terminal of the shunt are soldered to the board (to the footprint that was formerly used for the power inductor), and the other two terminals are connected a few centimeters above the board.

In addition to the two ZCU104 boards with the Zynq UltraScale+ XCZU7EV MPSoC, we used a Xilinx ZCU102 board with the Zynq UltraScale+ XCZU9EG MPSoC, i.e., a different member (EG instead of EV) of the UltraScale+ MPSoC family without video codec unit but therefore with more LUTs, on-chip memory, and DSP blocks. We ported our entire design to that third board and deployed our ML-based SCA detection method (trained for the two ZCU104 boards) on the ZCU102 board without retraining it. We wanted to see whether our ML model can be successfully used on a board type and FPGA type that was not used for training at all.



(a) Removed fan, heat sink, and heat spreader



(b) Shunt resistor inserted on the board

Fig. 3: Tampering the ZCU104 boards for (a) EMA attacks and (b) PA attacks

As measurement load for detecting EMA attacks, we used 2500 ring oscillators (ROs), where each RO is implemented by a single LUT. For detecting PA attacks, we used 25000 ROs, and as background load, we used 15 blocks with 1000 ROs each, where each block could be enabled separately. All ROs are implemented with the required constraints to avoid their removal by the synthesis tools (i.e., DONT_TOUCH and ALLOW_COMBINATORIAL_LOOPS).

For the measurement scenarios to train and evaluate our ML-based SCA detection, we varied the following parameters:

- Background Load: varied from 0 to 15 enabled blocks with a step size of 1.
- Ambient Temperature: varied between 24 °C and 90 °C with a step size of 2 °C (between 24 °C and 72 °C) and 6 °C (between 72 °C and 90 °C).
- Fan Speed: varied from 0 % to 100 % with a step size of 20 %.

In preparation of a measurement, we configured all parameters and waited for 10 s without having the measurement load enabled. We then enabled the measurement load for 30 s and afterwards had a 10 s recovery phase before we moved to the next scenario. Overall, we used 840 scenarios for EMA and split them into 672 scenarios for training and 168 scenarios for testing. For PA, we used 110 scenarios (note that the ambient temperature is hardly relevant for PA) and split them into 88 scenarios for training and 22 scenarios for testing. The readings are taken from SysMon with 5.2 MHz [26].

B. Performance of the classifiers

After the measurements and building the training- and the test datasets, the next step is to train the different models and to evaluate their performance. The classifiers mentioned in Section IV are all trained and the results of evaluating their performance are shown in Table I. The evaluation is based on

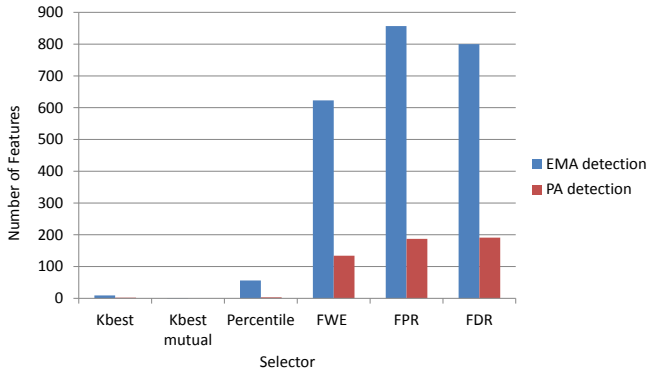


Fig. 4: Number of needed features for maximum accuracy of each selector

the accuracy and precision of each classifier for each attack (PA and EMA). Additionally, the number of features required for each classifier are shown. The *accuracy* metric is defined as the ratio of correct classification decisions from all the classification decisions. The *precision* metric is defined as the ratio of the correct tampered classification decisions from all the tampered classification decisions (correct and wrong).

As it can be seen in Table I, most of the classifiers perform well for both attack types and need at most two features. Only the classifiers NuSVC and GNB performed significantly worse than the others. Moreover, the DT classifier had the best performance in terms of accuracy and precision for both attack types. The good performance of DT is in line with the observations of ref. [44] that states that tree-based classifiers perform well for problems that need to distinguish between two classes.

It can also be seen in Table I that the EMA attack can be detected with higher accuracy and precision than the PA attack. The reason is that for the PA attack, a very small resistance of only $15\,\mathrm{m\Omega} = 0.015\,\Omega$ was used as measurement shunt. Therefore, the supply voltage fluctuation is not as significant as if a bigger resistor would have been used. But that is not possible when attacking systems with a small supply voltage ($<1\,\mathrm{V}$). When using a larger resistance, then the larger voltage drop would make the system operate outside its specification, which typically leads to a reset or an undefined state. So, for systems that operate with $<1\,\mathrm{V}$, it is hard to perform a PA attack and it is also hard to detect such an attack. As we will see in Section V-D, many other PA detection methods are limited to older chip technologies, where $\sim 1\,\Omega$ resistances are used. But with our approach, we are even able to detect PA tampering for $15\,\mathrm{m\Omega}$ resistances.

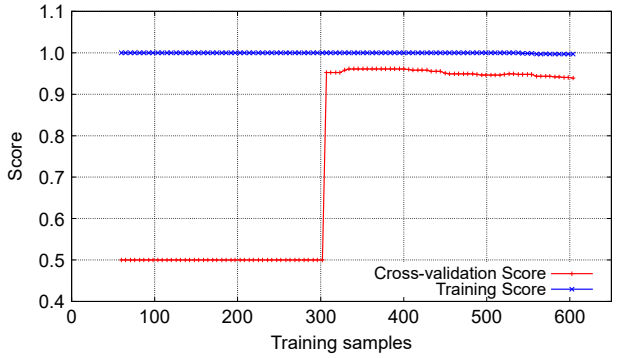
So we chose DT as classifier and implemented it on the FPGA-based SoC, as in addition to its good performance, it is easy to implement and has a small overhead, as we will see. Additionally, DT was tested for portability on a third untampered FPGA board of a different type (ZCU102) without retraining for that board. There, its accuracy was 0.952 for EMA and 0.83 for PA, which shows that our ML-based SCA detection method is not limited to the systems and scenarios used for training.

We now analyze the performance of DT deeper, as it is the most relevant classifier for this work. To get the best performance out of it, we evaluate the different feature selectors

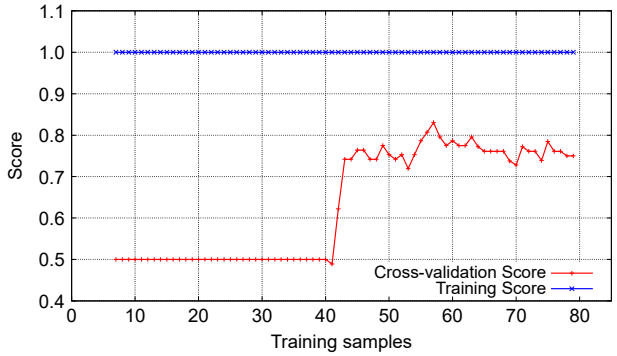
TABLE I: Performance of the different ML Models. The accuracy and precision reported here are based on testing 22 samples for PA and 168 for EMA.

Machine Learning Model	Accuracy PA	Accuracy EMA	Precision PA	Precision EMA	# Features PA	# Features EMA
Decision Tree	0.9090	0.9880	0.9090	0.9883	2	1
Extra-Trees	0.8636	0.9880	0.8928	0.9883	1	1
Support Vector Classification	0.8181	0.9821	0.8290	0.9827	1	1
Random Forest	0.8636	0.9880	0.8928	0.9883	1	1
Number-controlled Support Vector Classification	0.7727	0.8988	0.7750	0.9057	1	1
K-Nearest Neighbors	0.8181	0.9880	0.8290	0.9883	1	2
Gaussian Naive Bayesian	0.7727	0.9404	0.7750	0.9414	1	1
Adaptive Boosting	0.8636	0.9821	0.8928	0.9827	1	1

mentioned in Section IV to choose the features that provide the highest accuracy. Figure 4 shows the number of features used by each selector. As it can be seen, the Fwe, Fpr, and Fdr selectors need more than 400 features for EMA and more than 100 features for PA. This means that for this model, optimizing for a specific error requires a significant increase of the features while the overall accuracy is not improved. This would translate into high performance- and/or area overhead to extract all the features. Therefore, these three selectors are excluded and we focus on KBest and Percentile. Based on the evaluation of the selectors, one feature is used for EMA detection, which is the spectral Welch density. As for the PA detection, two features are selected and both calculate an aggregated linear trend with different chunk sizes.



(a) Cross-validation EMA



(b) Cross-validation PA

Fig. 5: Cross-validation results showing the needed training samples to avoid overfitting

The final step is to perform cross-validation to fine-tune the number of samples needed to train the model. It is done to make sure that the model is not overfitted. Figure 5 shows the training samples needed for each attack type. The figure shows that to get reliable classification, 300 training samples are needed for EMA. As for PA, between 40 to 60 samples are needed to get reliable data. EMA requires more samples as it has more variables that affect its state (the fan speed and the ambient temperature). These two results show that our model, which is trained with 672 samples and 88 samples for EMA and PA respectively, is not overfitted.

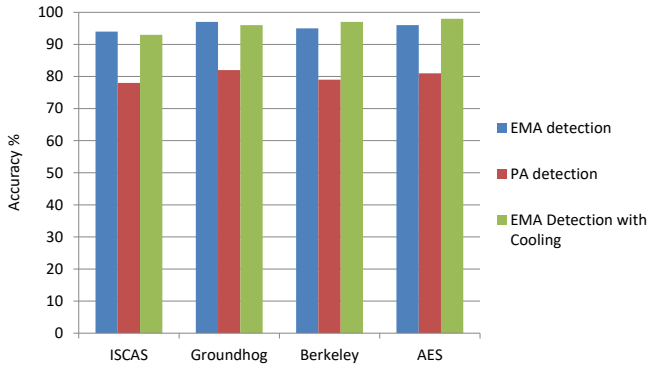


Fig. 6: Accuracy of the trained classifier with different background loads from different benchmarks.

We extend the evaluation to cases where the background load is not only ROs. To mimic a more realistic behavior, we replace the background load with benchmark circuits from the ISCAS, Groundhog, and Berkeley benchmarks and open source AES implementation [45]–[48]. They are used in inference only but not in training. For each benchmark, we run 100 experiments for PA detection, 100 for EMA detection and EMA detection with extra cooling using external fan to compensate for the removed cooling. All the runs are done in ambient temperature with no changes to the parameters. Figure 6 shows the results which agree with the results from the cross-validation. Neither the different benchmarks nor the cooling affected the detection accuracy and we see that again the accuracy for EMA is higher than for PA.

We could not study the robustness of our approach against aging due to a lack of differently aged chips. Aging might affect sensor readings or power consumption, but we expect minor effects compared to process variation since FPGA SoCs are not used in high voltage and high temperature conditions. We tested our approach with multiple FPGA boards and different sizes (ZCU102 and ZCU104) without retraining the ML model. Moreover, our method proved robust, distinguishing benign changes (e.g., ambient temperature) from attacks, assuring us it would handle aging scenarios effectively.

TABLE II: Utilization report of measurement and classification design on the Zynq UltraScale+ MPSoC Design derived from Vivado IDE

Module Name	CLB LUTs	CLB Registers	CLB	LUT as Logic	For Detection
AXI PL Interface	7811 (3.39%)	7037 (1.53%)	1209 (4.19%)	7422 (3.22%)	true
Background Load	8622 (3.74%)	0 (0%)	1286 (4.47%)	8622 (3.74%)	
Measurement Load EMA	1469 (0.64%)	0 (0%)	214 (0.74%)	1469 (0.64%)	true
Measurement Load PA	13434 (5.83%)	0 (0%)	1926 (6.69%)	13434 (5.83%)	true
PWM Fan Control	59 (0.03%)	44 (¡0.01%)	14 (0.05%)	59 (0.03%)	
Total usage Detection	21245 (9.22%)	7037 (1.53%)	3135 (10.88%)	20856 (9.05%)	true
Total usage Training	29926 (12.99%)	7081 (1.54%)	4435 (15.4%)	29537 (12.82%)	

C. Performance Overhead

The detection system proposed in this work has parts implemented in PL and other parts in PS. Therefore, in order to evaluate the overhead to the main system, two evaluations have to be done. The first evaluation shows the hardware resources needed on the PL side in Table II. The "Background Load" and the "PWM Fan Control" (cf. Figure 2) are only needed for the training data collection and are excluded from the final system. The "Measurement Load" and the "AXI PL Interface" are both used in the final system. As the AXI PL Interface is needed for any design that uses both PL and PS, the measurement load is the only part that has to be added for our detection solution. It has a reasonable resource usage of $5.83\,\%$ of the LUTs for PA detection. For EMA detection, even less resources of only $0.64\,\%$ are needed. For a system to detect both, PA and EMA, the measurement load for EMA is implemented by enabling a subset of the PA measurement load, leading to no extra hardware overhead for EMA detection. As the measurement load is implemented on the PL, the partial reconfiguration feature could potentially be used to reconfigure the measurement load only when it is actually needed. As an attack needs tampering of the system, which is not easily reversible, our detection solution can be active at fixed time intervals instead of always running it. When it is not used, the area of the measurement load may be used for different user-specific applications, thus minimizing the resource utilization. However, that is beyond the scope of this work and may not be needed in many cases.

The second evaluation shows the PS timing overhead in Table III. Our ML-based SCA detection performs two main operations. At first, the measurement is performed by enabling the measurement load, which dominates the timing requirement as it runs for $30\,\mathrm{s}$. Next, the classifier is executed, which consists of two parts: feature extraction and classification. For EMA, the feature extraction takes $122\,\mathrm{ms}$ and for PA it takes $40\,\mu\mathrm{s}$. The reason for the significantly longer time of the EMA feature extraction is that calculating the spectral Welch density is more complex than calculating the aggregated linear trend. The execution time for the DT classifier is only $0.5\,\mu\mathrm{s}$ for both attack types. This is expected, as DTs have lightweight implementations. The classification overhead is shown in Table III. This overhead occurs on the PS. In addition, the RO detection on the PL requires 30 seconds. But this RO data collection does not affect the performance of applications running on the PS.

D. Comparison-to-the-state-of-the-art

Finally, we evaluate our system by comparing how good it performs against the state of the art. For comparison, we focus on state-of-the-art PA detection, because state-of-the-art EMA detection methods assume systems that cannot be compared to ours. For instance, they use LC sensors that are not available in COTS systems, or they target CPU-centric desktop computers that provide very different attack vectors and detection opportunities, as explained in Section III.

TABLE III: Classification Time Overhead

	Features Selected	Feature Extraction Overhead	Classification Overhead
PA	aggregated linear trend	$40\,\mu\mathrm{s}$	$0.5\,\mu\mathrm{s}$
EMA	density spectral Welch function	$122\,\mathrm{ms}$	$0.5\,\mu\mathrm{s}$

Table IV shows the comparison that is based on a similar comparison shown in [7]. All three state-of-the-art works shown in the table provide similar metrics that we can compare to. Two of them use ML and PDN voltage variation to detect PA attacks [5], [8] and one uses the frequency shift of ROs to detect PA attacks. The works that use ML need significantly more features than our solution. Additionally, our method supports a significantly smaller measurement shunt resistance, which means that it can also be used for modern technologies, as explained in Section V-B. Ref. [5] does not report the lowest resistance value that they can detect and [7] does not report a specific value, but only states they can detect less than $1\,\Omega$. Moreover, they state that the minimum resistance value is highly limited by the thermal variations that could shift the frequency of the ROs; otherwise they would have a high false-positive rate. Our solution has the second-lowest area requirement; both [5], [8] have higher area requirement than our solution as they need an ADC for each sensor which needs a significant area overhead.

Only the solution proposed in [8] has a higher accuracy than our solution. However, this comes at the cost of having a significantly higher area requirement compared to all other solution. Note that [7] does not explicitly mention their accuracy, rather they mention again their trade-off between high false positive rate and detection of 'small' resistance values. Our solution can target the latest technology in comparison to the other works, making it suitable for the newest series of devices.

Additionally, it needs to be noted that all three related works are solely based on simulations, whereas our solution is completely implemented and tested in hardware. Simulations usually cannot match the real implementation, e.g., [5], [8] use ideal models for voltage distribution and the accuracy of the multiple ADCs inserted to the PDN, whereas in a real implementation they would have to face the noise and variations of a real system (e.g., added capacitance which cannot be evaluated with ideal devices), which would affect the whole evaluation of their methods. Hence, without considering

TABLE IV: Comparison of our solution with state of the art, based on results shown in [7]

Approach	**ML-based [8]**	**ML-based [5]**	**RO-based [7]**	**Our Solution**
Calculation time	6.6 μs	394 ns	2 μs	40.5 μs
Detection metric	PDN variation	PDN variation	ΔPhase & ΔN_{rising}	supply voltage variation
Min SCA_R	1 Ω	N.A.	$< 1\,\Omega$	15 mΩ
Technology	45 nm	45 nm	22 nm	16 nm
Additional ADCs	Needed	Needed	Not Needed	Not Needed
Area (kGE)	44444.44	749.62	1.98	107.47
# Features	110	30	N.A.	2
Accuracy	0.98	0.8	N.A.	0.9090
Evaluation	simulation only	simulation only	simulation only	hardware tested

the effects of inserting several of non-ideal ADCs to the PDN, the accuracy reported by these works cannot be trusted. The solution presented in [7] uses a more practical model with thermal effects, still, real-world chip behavior can lead to lower accuracy. This is especially relevant as they are using ROs that can deviate from their simulations, e.g., based on process variation while fabricating the chip, etc.

The main drawback of our work compared to the state of the art is the detection time. Our measurement and feature extraction come with a noticeable time overhead. However, this has a negligible effect in real-world systems, because (i) tampering takes time and cannot be changed quickly, and (ii) side-channels attack require many measurements (in the range from several 100 thousand to several million encryptions that need to be observed). Hence, while slower, our solution can still detect tampering efficiently.

Summarizing: The ML-based methods from [5], [8] require many new sensors and ADCs to be inserted on the power distribution network (needs a special ASIC design) to precisely detect the fluctuation. For larger chip designs, the number of needed sensors and ADC would even increase. Subsequently, the area overhead will also increase. Using ROs as the sensor as in [7] is not very reliable and would lead to many false positives, depending on the ambient temperature, the background load, etc. These two main shortcomings are overcome by our solution, as the ROs are not used as sensors, but are only used to generate a load response for the supply voltage and temperature sensors. Moreover, this significant change in the behavior makes it possible to detect the attack without the need of multiple sensors on the PDN itself.

VI. Conclusions

In this paper, we presented our novel concept that uses machine learning (ML) with the goal to detect whether or not a system was tampered for a power analysis (PA) or electromagnetic analysis (EMA) side-channel attack (SCA), as these two attack types are the most prominent SCAs in the area of non-invasive physical attacks against embedded systems that are deployed in the field and may be under complete control of an attacker. As it is generally challenging to successfully distinguish an actual manipulation for an attack from a legitimate scenario (e.g., operating in a hotter ambient environment), we tested several typical machine learning methods with different parameters. We also evaluated their performance with regards to accuracy (i.e., number of correctly classified systems relative to all systems), precision (i.e., number of systems correctly classified as tampered relative to all systems classified as tampered), area overhead, and computation overhead.

The main idea is to use a load generator (in our case: ring oscillators) and to measure how the system reacts on load changes with regards to input voltage fluctuations and on-chip temperature measurements, to decide whether or not the system was tampered. For detecting EMA (PA) attacks, we reach a high accuracy of 0.9880 (0.9090) and a high precision of 0.9883 (0.9090). This comes at a reasonable resource overhead of $<1\,\%$ ($\sim 6\,\%$) of the available LUTs and an on-chip classification time of only $\sim 122\,\mathrm{ms}$ ($\sim 40\,\mu\mathrm{s}$). The entire system including the on-chip measurement infrastructure, the ML-based classification, and the tampering for PA and EMA attacks are implemented and evaluated on FPGA boards. Even though we focus our implementation and evaluation on FPGA-based SoCs, our concept is not generally limited to such systems. It can be applied to other systems as well, as long as they provide a power supply sensor, a temperature sensor, a load generator, some memory to buffer the sensor readings, and a CPU to execute the ML classifier in software or a (reconfigurable) hardware to execute it in hardware. In comparison to the state-of-the-art we offer a reduction of the area overhead, detect two orders of magnitude insertion of probes, and offer actual implementation results in contrast to the simulations based on ideal circuits which are not accurately modelling the circuit and noise. Moreover, our tool can detect tampering for both PA and EMA.

Acknowledgment

The authors thank Carina Kübler and Varun Manjunath for their help in implementing the detection system.

References

[1] A. Mosenia, S. Sur-Kolay, A. Raghunathan, and N. K. Jha, "Wearable medical sensor-based system design: A survey", *IEEE Transactions on Multi-Scale Computing Systems (TMSCS)*, vol. 3, no. 2, pp. 124–138, 2017.

[2] T. Gomes, S. Pinto, A. Tavares, and J. Cabral, "Towards an FPGA-based edge device for the Internet of Things", in *Conference on Emerging Technologies & Factory Automation (ETFA)*, 2015, pp. 1–4.

[3] Y.-k. Choi, J. Cong, Z. Fang, Y. Hao, G. Reinman, and P. Wei, "A quantitative analysis on microarchitectures of modern CPU-FPGA platforms", in *Design Automation Conference (DAC)*, 2016, p. 1–6.

[4] M. Elnawawy, A. Farhan, A. Al Nabulsi, A. Al-Ali, and A. Sagahyroon, "Role of FPGA in Internet of Things applications", in *International Symposium on Signal Processing and Information Technology (ISSPIT)*, 2019, pp. 1–6.

[5] F. Kenarangi and I. Partin-Vaisband, "Exploiting machine learning against on-chip power analysis attacks: Tradeoffs and design considerations", *IEEE Transactions on Circuits and Systems I: Regular Papers (TCAS-I)*, vol. 66, no. 2, pp. 769–781, 2019.

[6] J. González-Gómez, K. Cordero-Zuñiga, L. Bauer, and J. Henkel, "The first concept and real-world deployment of a gpu-based thermal covert channel: Attack and countermeasures", in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2023, pp. 1–6.

[7] N. Gattu, M. N. I. Khan, A. De, and S. Ghosh, "Power side channel attack analysis and detection", in *International Conference On Computer Aided Design (ICCAD)*, 2020, pp. 1–7.

[8] D. Utyamishev and I. Partin-Vaisband, "Real-time detection of power analysis attacks by machine learning of power supply variations on-chip", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 39, no. 1, pp. 45–55, 2020.

[9] G. Camurati, S. Poeplau, M. Muench, T. Hayes, and A. Francillon, "Screaming channels: When electromagnetic side channels meet radio transceivers", in *Conference on Computer and Communications Security (CCS)*, 2018, p. 163–177.

[10] Y. Han, S. Etigowni, H. Liu, S. Zonouz, and A. Petropulu, "Watch me, but don't touch me! contactless control flow monitoring via electromagnetic emanations", in *Conference on Computer and Communications Security (CCS)*, 2017, p. 1095–1108.

[11] N. Khan, S. Nitzsche, R. Frank, L. Bauer, J. Henkel, and J. Becker, "Amplifying side channel leakage by hardware modification of xilinx zynq-7 fpga evaluation boards", in *Proceedings of International Conference on Emerging Security Information, Systems and Technologies*, 2019.

[12] P. Maistri, S. Tiran, P. Maurine, I. Koren, and R. Leveugle, "Countermeasures against EM analysis for a secured FPGA-based AES implementation", in *International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, 2013, pp. 1–6.

[13] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks", in *International Cryptology Conference (CRYPTO)*, 1999, pp. 398–412.

[14] S. Skorobogatov, "Physical attacks and tamper resistance", in *Introduction to Hardware Security and Trust*, M. Tehranipoor and C. Wang, Eds. Springer, 2012, pp. 143–173.

[15] R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, 2nd ed. John Wiley & Sons, 2008, ch. Physical Tamper Resistance, p. 483–521.

[16] *Anti-Tamper Capabilities in FPGA Designs*, Altera Corporation, 7 2008, white Paper WP-01066-1.0.

[17] J. Knechtel, S. Patnaik, M. Nabeel, M. Ashraf, Y. S. Chauhan, J. Henkel, O. Sinanoglu, and H. Amrouch, "Power side-channel attacks in negative capacitance transistor", *IEEE Micro*, vol. 40, no. 6, pp. 74–84, 2020.

[18] J. Gonzalez-Gomez, L. Bauer, and J. Henkel, "Cache-based side-channel attack mitigation for many-core distributed systems via dynamic task migration", *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 2440–2450, 2023.

[19] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The EM side-channel(s)", in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2002, pp. 29–45.

[20] H. Nassar, P. Machauer, D. R. E. Gnad, L. Bauer, M. B. Tahoori, and J. Henkel, "Covert-hammer: Coordinating power-hammering on multi-tenant fpgas via covert channels", in *Proceedings of the 2024 ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 43. [Online]. Available: https://doi.org/10.1145/3626202.3637613

[21] J. González-Gómez, M. B. Sikal, H. Khdr, L. Bauer, and J. Henkel, "Smart detection of obfuscated thermal covert channel attacks in many-core processors", in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023, pp. 1–6.

[22] J. Waddle and D. Wagner, "Towards efficient second-order power analysis", in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2004, pp. 1–15.

[23] Y. Hori, T. Katashita, A. Sasaki, and A. Satoh, "Electromagnetic side-channel attack against 28-nm FPGA device", *Pre-proceedings of International Workshop on Information Security Applications (WISA)*, 2012.

[24] A. Vasselle, P. Maurine, and M. Cozzi, "Breaking mobile firmware encryption through near-field side-channel analysis", in *Workshop on Attacks and Solutions in Hardware Security (ASHES)*, 2019, pp. 23–32.

[25] A. Barenghi and G. Pelosi, "Side-channel security of superscalar CPUs: Evaluating the impact of micro-architectural features", in *Design Automation Conference (DAC)*, 2018, pp. 1–6.

[26] *UltraScale Architecture System Monitor User Guide (UG580)*, Xilinx, Inc., 2021.

[27] T. E. Anderson, "The performance of spin lock alternatives for shared-memory multiprocessors." *IEEE TPDS*, 1990.

[28] E. Peterson, *Developing Tamper-Resistant Designs with UltraScale and UltraScale+ FPGAs Application Note (XAPP1098 v1.4)*, Xilinx, Inc., 2021.

[29] D. Wang, Z. Qian, N. Abu-Ghazaleh, and S. V. Krishnamurthy, "PAPP: prefetcher-aware prime and probe side-channel attack", in *Design Automation Conference (DAC)*, 2019, pp. 1–6.

[30] C. Liu, C. Yang, and Y. Shen, "Leveraging microarchitectural side channel information to efficiently enhance program control flow integrity", in *International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2014.

[31] Z. Dyka, C. Walczyk, D. Walczyk, C. Wenger, and P. Langendoerfer, "Side channel attacks and the non volatile memory of the future", in *International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES)*, 2012, p. 13–16.

[32] M. Mushtaq, A. Akram, M. K. Bhatti, M. Chaudhry, M. Yousaf, U. Farooq, V. Lapotre, and G. Gogniat, "Machine learning for security: The case of side-channel attack detection at run-time", in *International Conference on Electronics, Circuits and Systems (ICECS)*, 2018, pp. 485–488.

[33] A. Dhavlle, S. Bhat, S. Rafatirad, H. Homayoun, and S. M. PD, "Work-in-progress: Sequence-crafter: Side-channel entropy minimization to thwart timing-based side-channel attacks", in *International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES)*, 2019.

[34] *Zynq UltraScale+ MPSoC Embedded Design Methodology Guide*, Xilinx, Inc., 2017, user Guide UG1228 (v1.0).

[35] S. Ravi, A. Raghunathan, and S. Chakradhar, "Tamper resistance mechanisms for secure embedded systems", in *International Conference on VLSI Design*, 2004, pp. 605–611.

[36] N. Homma, Y.-i. Hayashi, N. Miura, D. Fujimoto, D. Tanaka, M. Nagata, and T. Aoki, "EM attack is non-invasive?-design methodology and validity verification of EM attack sensor", in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2014, pp. 1–16.

[37] H. Nassar, S. Pankner, L. Bauer, and J. Henkel, "Late breaking results: Configurable ring oscillators as a side-channel countermeasure", in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023, pp. 1–2.

[38] A. Singh, M. Kar, S. Mathew, A. Rajan, V. De, and S. Mukhopadhyay, "Exploiting on-chip power management for side-channel security", in *Design, Automation & Test in Europe (DATE)*, 2018, pp. 401–406.

[39] A. Aysu, M. Orshansky, and M. Tiwari, "Binary Ring-LWE hardware with power side-channel countermeasures", in *Design, Automation & Test in Europe (DATE)*, 2018, pp. 1253–1258.

[40] L. Goubin and J. Patarin, "DES and differential power analysis the "duplication" method", in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 1999, pp. 158–172.

[41] D. Jayasinghe, A. Ignjatovic, J. A. Ambrose, R. Ragel, and S. Parameswaran, "QuadSeal: Quadruple algorithmic symmetrizing countermeasure against power based side-channel attacks", in *International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, 2015, p. 21–30.

[42] C. Kasmi, J. Lopes-Esteves, and M. Renard, "Autonomous electromagnetic attacks detection considering a COTS computer as a multi-sensor system", in *URSI General Assembly and Scientific Symposium (URSI GASS)*, 2014, pp. 1–4.

[43] J. Minker, *Logic-based artificial intelligence*. Springer Science & Business Media, 2012, vol. 597.

[44] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *Journal of Machine Learning Research*, vol. 15, no. 90, pp. 3133–3181, 2014.

[45] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits", in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 1989.

[46] P. Jamieson, T. Becker, P. Y. K. Cheung, W. Luk, T. Rissa, and T. Pitkänen, "Benchmarking and evaluating reconfigurable architectures targeting the mobile domain", *ACM Transactions on Design Automation of Electronic Systems (TODAES))*, 2010.

[47] J. Pistorius, M. Hutton, A. Mishchenko, and R. Brayton, "Benchmarking method and designs targeting logic synthesis for fpgas", in *International Workshop for Logic Synthesis (IWLS)*, 2007.

[48] P. Maene and I. Verbauwhede, "Single-cycle implementations of block ciphers", in *Lightweight Cryptography for Security and Privacy*, 2016.