Design of side-channel resistant power monitors

Davide Zoni*, *Member, IEEE*, Luca Cremona, and William Fornaciari, *Senior Member, IEEE*

***Abstract*—In modern computing platforms, power monitors are employed to deliver online power estimates to support different run-time power-performance optimization methodologies. However, the possibility of setting up a successful side-channel attack by analyzing the power estimates imposes the use of a suitable and systematic approach in the design of such power monitors. This paper proposes a design methodology to automatically identify and implement side-channel resistant power monitors at the hardware level, for generic computing platforms. The methodology works by designing a power monitor for which the switching activity of the signals used to compute the power estimates is not a function of both the secret key and the plaintext/ciphertext values processed by the computing platform. According to the most recent standardized methodologies to assess the side-channel security, our experimental validation leverages both CPA and t-test analysis considering a general purpose System-on-Chip executing different cryptographic primitives and an application-specific accelerator implementing the AES-128 algorithm. Our results confirm the impossibility of retrieving the secret key from the power estimates provided by our side-channel resistant power monitor. Considering several temporal resolutions, we highlight an accuracy error of the power estimates limited to less than 2.7%, as well as an average area and power overheads for the protected power monitors lower than 6% and 5%, respectively. To this end, the proposed methodology is able to deliver a side-channel resistant power monitor within state-of-the-art accuracy error and overheads.**

***Index Terms*—Power monitoring, side-channel attacks, hardware security, computer architecture.**

I. Introduction

Over the last two decades, side-channel attacks demonstrated a large variety of ways of exfiltrating sensible information from a computing platform [1]. In general, the *side-channel attacks* (SCAs), also known as passive, i.e., non-invasive, attacks, can extract sensible information by exploiting the correlation between the data being computed and one or more environmental signals generated by the computing platform. Depending on the attack methodology, side-channel attacks are split in two classes: *i) emission-based*, and *ii) microarchitectural*. Emission-based SCAs require physical proximity to the target device without the need of executing any spy application on such target. These attacks measure the execution time, or the power consumption, or the electromagnetic field of the victim device without permanently altering it, and therefore they leave no evidence of an attack behind. In contrast, microarchitectural SCAs do not require physical proximity to the target platform since they leverage malicious or cooperating spy applications that are co-located/executed on the same platform with the victim application. In this work, we address the security vulnerability due to the use of the online power monitors, that offer a novel and effective way to implement a microarchitectural side-channel attack. Online power monitors are used to support run-time techniques to optimize the energy efficiency of the computing platform. They are the de-facto solution to deliver a periodic estimate of the power consumption of the monitored computing platform. More specifically, the power estimate is obtained by leveraging the relationship between the power consumption and the internal switching activity of the target device. However, the switching activity has been demonstrated to be a rich source of information that allow successful side-channel attacks [2].

The realization of a power monitor encompasses two main steps. First, the *power model identification* step finds out a mathematical relationship between the power consumption and the switching activity of the computing platform. Second, the *power monitor design* stage physically implements the identified power model into the target platform. The possibility of exploiting such relationship at different abstraction levels spurred the evolution of both software- and hardware- implemented power monitors, where each of them offers a different trade-off between the accuracy of the power estimates and the performance and area overheads. Software power monitors leverage the values of the high-level platform statistics related to the executed applications that are usually made available by means of the architectural performance counters. Such statistics measure the number of different architectural events, e.g., number of cache misses, or idle CPU cycles within the observed period of time. Software power monitors are implemented as software applications and they are primarily intended to support off-the-shelf computing platforms, for which any change of the microarchitecture is impossible. To the best of our knowledge, the state-of-the-art contains no research related to the side-channel vulnerability of software implemented power monitors. However, several state-of-the-art proposals highlight the possibility of setting up a successful side-channel attacks targeting the architectural performance counters exploited to realize the software power monitors [3], [4]. In contrast, hardware power monitors leverage the switching activity at circuit level to deliver accurate power estimates at high temporal resolution, without affecting the performance of the computing platform. The use of the signals' switching activity at microarchitectural level to calculate the power estimate creates a strong link between the calculated power sample and the program data values driven onto the monitored signals. In this scenario, any side-channel resistant power monitor must carefully consider the side-channel information leakage of any signal and statistics before using it to compute

* Corresponding and leading author: Davide Zoni

D. Zoni, L. Cremona and W. Fornaciari were with the Dipartimento di Elettronica Informazione e Bioingegneria, Politecnico di Milano, 20133 Milano, Italy.

E-mail: {davide.zoni,luca.cremona,william.fornaciari}@polimi.it

Manuscript received April 22, 2019; revised XXX.

the power estimates.

Contributions - To the best of our knowledge, this is the first design methodology to deliver side-channel resistant power monitors also ensuring an accuracy error as well area and power overheads aligned to the ones of state-of-the-art unprotected power monitors. In particular, the contribution to the state-of-the-art is two-fold:

- **Side-channel resistant power modeling -** We present a novel power modeling strategy that allows to deliver a side-channel resistant power monitor against first-order SCAs. To enforce the side-channel resistance property, the power monitor makes use only of signals for which the corresponding switching activity is not a function of the secret key and of the plaintext values. The proposed approach neither hides nor shadows the power estimate by means of random architectures for which additional hardware resources and security proof are required upon the implementation. To demonstrate the generality and feasibility of our solution, the reported results consider power monitors targeting a general-purpose System-on-Chip executing software cryptosystems and a hardware accelerator implementing the AES-128 cryptosystem. The experimental validation, exploiting state-of-the-art correlation power analysis (CPA) techniques as well as the non-specific statistical t-test, confirms that it is impossible to retrieve any portion of the secret key from the power estimates calculated by our protected power monitor.
- **Accurate and effective power monitor -** Our solution is orthogonal to any power monitoring design methodology. To this end, it can be added to any power monitoring design strategy to deliver a side-channel resistant power monitor. In particular, the proposed design methodology delivers a side-channel resistant power monitor showing area and power overheads as well as an accuracy loss aligned with state-of-the-art power monitoring design methodologies. Area, power and accuracy loss figures are obtained considering the design of a side-channel resistant power monitor for a general-purpose SoC and a cryptographic hardware accelerator. In particular, the metrics have been measured on each implemented power monitor considering a set of temporal resolutions for the computation of the power estimates, ranging from 100us to 500us. The overall area and power overheads are always less than 6% and 5%, respectively, with a maximum accuracy loss limited to 2.7%.

Threat model - The proposed methodology aims to deliver a side-channel resistant power monitor for generic computing platforms also ensuring an accuracy loss as well as area and power overheads aligned to state-of-the-art unprotected power monitors. Figure 1 depicts the addressed security scenario. The power monitor (`PwrMon`) is a hardware module that continuously collects the switching activity from a set of selected signals of the computing platform (`Computing platform`) and it periodically outputs the computed power estimate ($\hat{p}$). The attacker knows all the details of the implementation of the computing platform and of the power monitor, but has no access to the secret key. We note that in the case of a general-purpose computing platform, the attacker also knows the implementation details of the executed software-based cryptosystem. In the considered threat model, the attacker can collect the power traces generated by the power monitor, i.e., a series of $\hat{p}$ values over time, corresponding to the execution of the cryptosystem, and she/he is also allowed to execute properly chosen plaintext/chipertext attacks. The aim for the attacker is to retrieve the secret key of the executed cryptosystem by exploiting the output of the power monitor, i.e., the power estimates. In particular, the power estimates represent the unique side-channel signal accessible to the attacker. Such threat model fits the scenarios of microarchitectural-based side-channel attacks where, in contrast to traditional DPA attacks, the attacker has no need of a physical access to the target device. In fact, the run-time power consumption can be read out via a software/hardware interface when the attacker is logged-in, either locally or remotely, into the target computing platform. We note that the side-channel vulnerability of the power monitor lies in the actual statistics used to compute the power estimates. In particular, the use of the statistics coming from a single vulnerable signal to implement the power monitor either in software or in hardware can make it vulnerable to side-channel attacks. To this end, any power monitor for which the statistics selection process does not consider such security aspect is vulnerable to this type of side-channel attacks.

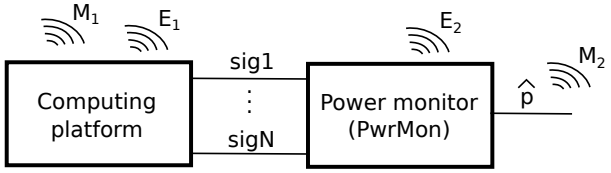


Fig. 1: Overview of the threats. The possible attacks can be micro-architectural (M_1, M_2) or emission-based (E_1, E_2), performed on the computing platform or the power monitor.

We note that traditional masking-based side-channel countermeasures are used to secure the computing platforms against side-channel attacks by changing the way the data are processed. However, a masked power monitor is useless against the discussed side-channel attack since the exploited vulnerability lies in the *value* of the power estimates of the monitored computing platform. We also note that the use of a masked computing platform avoids the need of designing side-channel resistant power monitors, since any computed power estimate will correlate with the masked version of the processed data. However, the high complexity and cost of designing masked computing platforms, especially for general purpose computing systems, motivates the use of side-channel resistant power monitors when the power estimates represent the unique side-channel source of information available to the attacker.

To this end, our methodology helps in delivering a power monitor for generic computing platforms, i.e., general-purpose SoCs and hardware accelerators, also ensuring that the power estimates cannot be exploited to retrieve the secret key, namely

to perform the M_2 side-channel attack. Moreover, our protected power monitor makes no use of vulnerable signals, thus its implementation is also theoretically resistant to possible emission-based side-channels attacks (E_2), but such evaluation falls outside the scope of this work. Finally, the proposed solution does not change the computing platform, thus it cannot mitigate emission-based (E_1) nor microarchitectural (M_1) side-channel vulnerabilities, if any.

Structure of the manuscript - The manuscript is organized in four parts. Section II discusses the background on the state-of-the-art on side-channel leakage detection methodologies and online power monitors. The proposed methodology to design a side-channel resistant power monitors is described in Section III. Experimental results are discussed in Section IV, while conclusions are drawn in Section V.

II. Background and related works

This section describes the link between the online power monitoring solutions and the side-channel attacks targeting the trace of power estimates as the side-channel signal. The security threat due to side-channel attacks against online power monitors is a well known problem. [5] presented a successful side-channel attack against an FPGA-implemented power monitor, without proposing any countermeasure. [6] discusses the use of a zero-mean noise adder to obfuscate the power estimates. However, the countermeasure is meant for smart meters used in civil electricity supply scenarios. The actual dynamics of the power consumption signal is several orders of magnitude slower with respect to the one of a computing platform and the electricity provider is interested in the actual energy consumption over a period of time, and not to the instantaneous value of the power consumption. Moreover, hiding techniques leveraging noise signals are known to be ineffective against side-channel attacks [7]. [3], [4] demonstrate the possibility of setting up a successful side-channel attack by exploiting the link between the statistics of the architectural performance counters and the data being computed. The presence of a link between the switching activity of selected hardware signals and the data being computed is a common knowledge [2].

The rest of this section has a two-fold objective. Section II-A focuses on the literature on power monitoring while a review on the side-channel leakage detection and exploitation methodologies is reported in Section II-B. We note that, to the best of our knowledge and apart from the already and partially on-topic discussed proposals, there are no state-of-the-art solutions targeting the design of a side-channel resistant power monitor for generic computing platforms.

A. On-line power monitoring

At run-time, the power consumption can be read out as either a *direct measurement* or an *indirect estimate*. The direct power measurement is achieved by means of analog sensors providing highly accurate power values at high temporal resolution. However, power meters suffer two limitations; *i)* the limited scalability prevents the deployment of more than few sensors even in complex computing platforms, and *ii)* the use of complex mixed analog-digital design methodologies to implement the power meter, increases the overall design complexity. In contrast, the indirect power estimate is produced by a power monitor exploiting an identified power model that is fed with the platform statistics at run-time. Feasibility, flexibility, and scalability make the use of power monitors a widely adopted best practice. The power model at the core of the power monitor leverages the relationship between the power consumption and the internal switching activity of the computing platform. In particular, the linear power model represents the de-facto standard family of models employed in state-of-the-art solutions due to its low overhead, high accuracy, and high achievable temporal resolution. Equation (1) defines the power estimate at time t ($\hat{p_t}$) as the weighted sum of the switching activity of selected signals of the computing platform ($s_{i,t}$). The power model identification process selects the signals and the corresponding time independent coefficients (c_i) to maximize the accuracy of the power estimates.

$$\hat{p_t} = \sum_i c_i \times s_{i,t} \tag{1}$$

The possibility of using the relationship between the switching activity and the power consumption at different abstraction levels motivates different power monitor implementations. Software-level power monitors make use of selected architectural performance counters to deliver a software implementation of the power model without changing the hardware implementation of the computing platform [8]–[10]. However, this flexibility comes at the cost of three drawbacks: *i)* a non-negligible performance overhead due to the software execution of the power model, *ii)* the need of architectural performance counters non always available, and *iii)* a relatively low accuracy due to the use of coarse grain and highly abstracted statistics. In contrast, hardware-level power monitors use the swithing activity of a selected subset of physical signals. The additional resource utilization and the need to change the hardware of the computing platform is balanced by a higher accuracy and temporal resolution with no performance overhead [11]–[14]. Moreover, the hardware-level statistics carry more information then architectural performance counters, thus motivating a vast research aiming at delivering compact hardware-level power monitors. To design a compact power model, [11] proposes a power monitoring methodology that only employs the control signal of the designs. [12], [13] propose to virtually split all signals into their single-bit representations and, then, to make use of the obtained single-bit signals to design the power model. In contrast, [14] highlights the importance of accounting for different types of switching activity depending on the considered signal. All these solutions target the design of highly accurate power monitors disregarding any security implication. In contrast, we note that the design of the power model must consider that *i)* some signals are driven directly or indirectly with critical program data, i.e., the secret key and the plaintext of the encryption procedure, and *ii)* the strategy employed to measure the switching activity can make the trace of power estimates vulnerable to side-channel attacks.

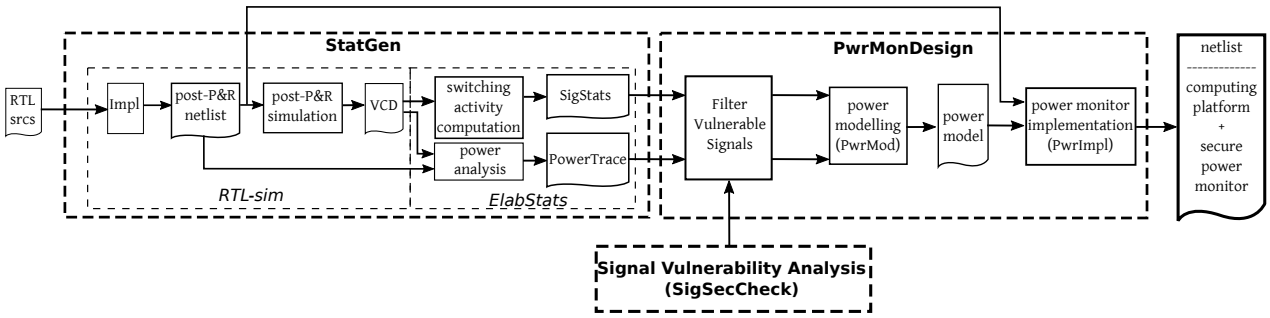


Fig. 2: The proposed design methodology toolchain to identify and to implement in hardware a side-channel resistant power monitor for generic Register-Transfer-Level (RTL) descriptions performing cryptographic computations.

B. Side-channel information leakage analysis

The classification adopted in the rest of this part distinguishes between the methodologies to *detect* the presence of information leakage and those capable to *exploit* it. In particular, the former highlight the potential vulnerability of a computing platform while the latter quantify the possibility of actually retrieving the secret key.

Side-channel leakage detection - The *Test Vector Leakage Assessment* (TVLA) methodology is a key element of the Cryptographic Module Validation Program (CMVP) [15], also highlighted in the FIPS-140-3 standard [16], to detect the presence of a side-channel information leakage in the target design. TVLA leverages the statistical Welch's t-test test targeting mean differences in carefully chosen partitions of trace measurements. In particular, the TVLA methodology can exploit the specific and non-specific statistical t-test test [17]. Non-specific TVLA partitions the collected measurements depending on public inputs. For example, the fixed-versus-random test is a non-specific t-test looking for a statistically significant difference between a trace set associated with a fixed plaintext input and another trace set associated with randomly varying inputs. In contrast, specific TVLA partitions the collected measurements depending on the value assumed by an intermediate key-dependent value computed by the analyzed cryptosystem implementation. [18] demonstrated that non-specific TVLA outperforms specific TVLA as the former highlights a lower number of false positives with respect to the latter. We note that both specific and non-specific statistical t-test tests are executed at any point in time, i.e., time-wise, of the collected trace measurements to draw conclusions about the overall vulnerability of the target implementation. However, apart from the risk of false positives, the interpretation of negative outcomes is difficult for TVLA without resorting to other evaluation tools to assess the side-channel resistance of the cryptographic primitive implementation [19]. The best practices to successfully leverage the use of leakage detection tests in security evaluation of cryptographic devices is discussed in [20]. We note that the statistical t-test test represent a primary tool to identify if the target computing platform is vulnerable to side-channel attacks.

Side-channel leakage exploitation - Once the TVLA detected a source of side-channel information leakage in the target design, the correlation power analysis (CPA) techniques can be used to retrieve the actual secret key. A typical CPA workflow is an instance of either a known plaintext or ciphertext attack against a symmetric cryptographic primitive and it is organized in four steps. First, we choose an intermediate value of the cipher computation which depends on a small portion of the key, usually either 1 or 8 bits, and a known attacker-controllable quantity, i.e., the plaintext. Second, the power consumption (the side-channel) is continuously measured during the execution of the cipher computation considering a large set of different, randomly distributed known plaintexts. Third, the attacker tries to predict the actual power consumption of the device employing the chosen *leakage model* which defines a mathematical formulation of the power consumption as a function of the secret key and the plaintext (or ciphertext). According to [21] even a reasonably coarse model of the power consumption of the target operation producing the intermediate value is sufficient to successfully attack the target. The Hamming Distance (HD) and the Hamming Weight (HW) are the two most popular leakage models to estimate data dependent power consumption. Considering the side-channel security domain, data dependent signals are those for which the switching activity is a function of both the secret key, i.e., the target information of the attack, and the plaintext/ciphertext, i.e., the quantity controllable by the attacker. The HD leakage model measures the number of single-bit switches of a signal between two consecutive clock cycles. Differently, HW measures the number of ones of a signal in a specific clock cycle as the power to load its fan out. The HW leakage model is simpler than the HD one since it only requires the knowledge of a single value driven on the signal. By leveraging the leakage model, the attacker can construct a hypothesized power consumption for each of the values of the attacked portion of the secret key. In the last step, each prediction is compared with the measured side-channel at each considered time instant by means of a statistical test. The correct value of the secret key portion is revealed as the prediction depending on it that will fit best the measurements. Starting from the work in [21], employing the Difference-of-Means (DoM) test to exfiltrate the secret key of a DES cipher, a large variety of other statistical distinguisher have been proposed. [22] investigates ways to enhance the accuracy of the differential power analysis taking also advantage of

more precise leakage models in the statistics. [2] employs the Pearson correlation coefficient to provide a comprehensive side-channel information leakage analysis of an in-order RISC CPU.

III. Methodology

This section discusses the design methodology to deliver a side-channel resistant power monitor for generic computing platforms (see Figure 2).

Key idea - the switching activity is the key for the accuracy and the security of the power monitor. From the accuracy viewpoint, the power monitor must employ the subset of signals of the circuit for which the switching activity ensures the best fit with the actual power consumption. From the security viewpoint, the switching activity is function of the data being processed and, thus, by inspecting the trace of the power estimates it is possible to retrieve the data driven on the signals used in the power monitor. The proposed methodology defines the concept of vulnerable signals as those signals for which the switching activity is a function of both the secret key, i.e., the target, unknown quantity to the attacker, and the plaintext/ciphertext, the controllable quantity by the attacker. From the attacker standpoint, the secret key is the sought information, while the plaintext is the controllable input to setup the side-channel attack. By avoiding the use of vulnerable signals to realize the power monitor, the power estimates cannot allow the retrieval of the secret key since the switching activity, that is at the core of the implied power model, does not depend on such information. Given the possible existence of a complex relationship between the switching activity, the secret key and the plaintext, our solution leverages the statistical t-test tests to highlight vulnerable signals. The experimental results demonstrate that such vulnerable signals are too few to determine a degradation of the accuracy if not considered in the realization of the power monitor. In contrast, such vulnerable signals are too many to guarantee that the implemented power monitor is secure against the presented side-channel attack if a security analysis is not part of the power monitor design methodology.

Toolchain overview - Figure 2 overviews the proposed flow as made of three stages, i.e., *i)* the `StatGen`, *ii)* the `SigSecCheck`, and *iii)* the `PwrMonDesign`. Starting from the Register Transfer Level (RTL) description of the target computing platform, the `StatGen` stage produces the power traces and the corresponding statistics for each signal in the design. We perform a post Place-and-Route (`post-P&R`) simulation of the target design to collect the Value Change Dump (VCD) file (see `RTL-sim` in Figure 2). The VCD file reports the variations, i.e., the switching activity over the time, for each signal in the simulated design. In particular, we leverage the VCD file to obtain the power traces (`PowerTrace`) and the switching activity (`SigStats`) statistics (see `ElabStats` in Figure 2). Starting from the whole set of collected statistics, i.e., the switching activity and the power traces, the signal vulnerability analysis stage (`SigSecCheck`) identifies the set of the physical signals of the platform that are vulnerable to side-channel attacks (see Section III-B for a detailed description). In particular, the `FilterVulnerableSignals` block removes the vulnerable signals and the corresponding statistics from the whole set of collected data. It is sufficient that a single bit of a signal is leaking information to discard such signal. The power monitor design stage (`PwrMonDesign`) delivers the netlist of the computing platform augmented with the netlist of the power monitor (`PwrImpl`) starting from the three received inputs: *i)* the set of switching activities of the signals for which no side-channel vulnerability has been observed in the `SigSecCheck` stage, *ii)* the power traces, and *iii)* the post Place-and-Route (`postP&R`) description of the target platform where the power monitor must be implemented. The rest of this section is organized in three parts. Section III-A describes the entire process of identifying the power model for generic computing platforms. Section III-B addresses the side-channel vulnerability analysis stage to filter out vulnerable signals. Section III-C details the implementation of the SCA-resistant power monitor starting from the identified power model.

Algorithm 1 Top-down hierarchical visiting algorithm.

```
1: function [model, e] EXPLOREHIERARCHY(M_0, e_Th)
2:     secureStats = [ ];
3:     for sig ∈ [M_0.I⃗ M_0.O⃗] do
4:         if SigSecCheck(sig) == 0 then
5:             secureStats.add(sig);
6:     [mId, e_M0] = ComputePwrModel(M_0, secureStats);
7:     if e_M0 < e_Th then
8:         model = mId; e = e_M0;
9:     else
10:        container = sortByPower(M_0.m_0 ... M_0.m_N);
11:        mIdList = [ ];
12:        for i = 1 : container.size do
13:            m_Tmp = container.pop(i);
14:            [mId, e] = exploreHierarchy(m_Tmp, e_Th);
15:            [mCont, eCont] = exploreHierarchy(container, e_Th);
16:            mIdList.add(mId);
17:            if compErr([mIdList mCont]) < e_Th then
18:                model = [mIdList mCont];
19:                e = compErr(model);
20:                break;
```

A. Power model identification

To obtain a compact yet effective power monitor, our power model identification strategy leverages the primary inputs and outputs to derive the power model. Such design strategy is meant to avoid modeling the complex non-linear relationships between the power consumption and the internal implementation of each module.

According to [14], [23], we implemented the `StatGen` block to deliver the switching activity in the form of both Single-Variation-Count (SVC) and Hamming-Weight-Count (HWC) for each signal in the design. The SVC measures a change in the signal, while the HWC measures the actual number of changed bits. For example, the transition of the 4-bit signal `foo` from 0001_2 to 0010_2 is registered as $SVC_{foo} = 1$ and $HWC_{foo} = 2$. We note that even if the

cross-correlation of the two statistics, i.e., SVC and HWC, measured from the same signal can be so strong to prevent their concurrent use in the same power model, they provide two alternative explanations of the power consumption for the same signal, thus enabling a potential improvement of the power model accuracy.

The power model identification algorithm employs a recursive approach to implement a top-down hierarchical visit of the target design (see the `ExploreHierarchy` function in Algorithm 1). The algorithm takes two inputs: *i)* the top module of the design (M_0), and *ii)* an upper bound that specifies the maximum accuracy error for the identified power model ($\mathbf{e_{Th}}$). The mathematical formulation of the identified power model (`model`) and the accuracy error ($\mathbf{e}$) represent the output of the algorithm. We note that the power model is a list of triples, where each triple is defined as the name of a signal of the design, the associated coefficient of the power model and the employed Switching Activity Counting Mode (SACM) that is selected as either the Single Variation Count (SVC) or the Hamming Weight Count (HWC). Equation (2) depicts the linear power model behind the proposed identification algorithm, where signals are accounted by measuring the switching activity either using the HWC mode or the SVC mode. We note that the time-dependent behavior of the power model in Equation (2) is due to the time-dependent switching activity collected from the *i*-th signal that contributes to the computation of the power estimates ($\hat{p_t}$) over time.

$$\hat{p_t} = \sum_i c_i \times s_{i,t}^{SVC} + \sum_j c_j \times s_{j,t}^{HWC} \quad (2)$$

Starting from the top module, Algorithm 1 initially performs a top-down visit of the target design hierarchy to find the best side-channel resistant power model within the accuracy upper bound $\mathbf{e}_{Th}$. As a first step, the `ExploreHierarchy` function filters out the signals for which a dependency between the switching activity and the program data being processed in the computing platform is detected, i.e., the vulnerable signals. We note that the violation of a single bit of a large signal is sufficient to filter out such signal (see lines $2-7$ in Algorithm 1). The remaining signal statistics (`secureStats`) are employed to identify the power model (`mId`) of M_0 (see line 8 of Algorithm 1). We note that the computed power model is accepted if its accuracy error is within the allowed error $\mathbf{e}_{Th}$ (see lines 9-11 of Algorithm 1). Otherwise, the children modules of M_0 are sorted in a descending order, according to their power consumption, and an iterative power model identification exploration starts (see lines 11-25 of Algorithm 1). At each iteration of the `for-loop` at line 14, the first sub-module in the sorted `container` list is popped out and its power model is identified (see mId at line 16 of Algorithm 1). Moreover, the remaining modules in the `container` list are identified within a single power model. To this extent, a bi-partition of the modules is created. The module identified in isolation is added to the $mIdList$ list, i.e., the list containing the power models identified on the modules consuming the majority of the power in the target design. At line 19, Algorithm 1 checks if the aggregate error of the power models in the $mIdList$ plus the power model identified for the $container$ modules is lower than the $\mathbf{e}_{Th}$ threshold. This approach allows to optimize the size of the final power model, since the iterative algorithm tries to identify a dedicated power model for the modules than contribute the most to the power consumption, while a single aggregate power model is identified for the remaining ones. The recursive visit of the target design terminates either with a set of identified power models or when the container list is empty and the error is bigger than the imposed $\mathbf{e}_{Th}$ threshold. We note that for each recursive call of the `ExploreHierarchy` function, the `SigSecCheck` task, for which a detailed description is provided in Section III-B, ensures that any identified power model makes only use of the switching activity from non-vulnerable signals.

Algorithm 2 Power model computation for module m.

```
1: function [model, e] ComputePwr(m, secureStats)
2:     e = MAXERR; model = [ ];
3:     for i = 1 : secureStats.size do
4:         C = (secureStats choose i);
5:         for j ∈ C do
6:             [model_new, e_new] = linReg(j, m.pwr);
7:             if e_new < e then
8:                 e = e_new; model = model_new;
```

The actual power model identification procedure for a module is described in Algorithm 2, i.e., the `computePwr` function. Algorithm 2 takes as input a module of the design (`m`) and the list of statistics pertaining to the primary inputs and outputs of such module. It is important to underline that the algorithm is fed with non-vulnerable statistics. The output of the algorithm is the identified power model (`model`) and the associated accuracy error ($\mathbf{e}$).

The two nested for-loops (lines 3 and 5) drive the exploration to favor the power models that require a small number of probed signals. For each iteration of the outer for-loop, the statement at line 4 determines the set of all the possible combinations of i signals. For each combination j, the inner for-loop (line 5) computes the power model ($model_{new}$) and the accuracy error (e_{new}) by means of a linear regression procedure (see line 8 in Algorithm 2).

The identified power model ($model_{new}$) becomes the new candidate if its accuracy error is smaller than the one of the current power model (see lines 7-9 in Algorithm 2). We note that, regardless the employed power model identification strategy, the side-channel vulnerability analysis allows to derive power models for which the corresponding power estimates are not function of the secret key and the plaintext values processed by the computing platform.

B. Side-channel vulnerability analysis

For each signal in the computing platform, the signal vulnerability analysis stage (`SigSecCheck` in Figure 3) tests the correlation between its switching activity and the processed data. The analysis makes use of the Test Vector Leakage Assessment (TVLA) methodology employing both specific and non-specific t-test statistical tests to assess the side-channel vulnerability of each signal. Such analysis is meant to seek evidence of sensitive data dependencies in the

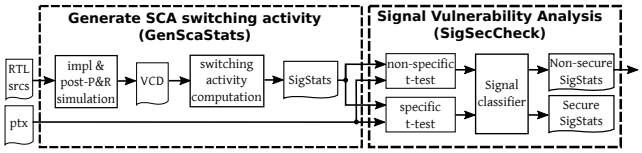


Fig. 3: Analysis flow used to assess the side-channel vulnerability of the physical signals of the target computing platform.

measured traces rather than actually retrieving sensitive information from the highlighted side-channel information leakage. Starting from the RTL description of the computing platform, we collect the switching activities (`SigStats`) from a set of post-implementation simulations of the target cryptosystem where each of them employs a different plaintext (see `ptx` in Figure 3).

We note that our side-channel vulnerability analysis only leverages the collected switching activity without requiring the use of power traces either estimated through the use of Electronic Design Automation (EDA) tools or directly measured from the prototyping platform. In fact, the power monitor makes use of the sole switching activity of selected signals to compute the power trace, i.e., the signal we want to protect. This is a crucial simplification that allows to greatly speed up the analysis. In fact, the use of estimated power traces can severely slow down the processing, i.e., by a factor of 100x, at least. Moreover, the use of measured power traces imposes the execution of the computing platform on a prototype board, thus severally delaying the design of the power monitor until when the computing platform reaches the prototype stage.

The specific t-test is employed to assess the relationship between a specific intermediate value computed by the cryptosystem and the measured switching activity of the signal under test. To this end, the selection of the intermediate value to consider is crucial to maximize the information of the statistical test. Algorithm 3 describes the implemented specific t-test to assess the correlation between the switching activity of a signal and the data being processed in the computing platform. Algorithm 3 takes six inputs: *i)* the round (`rnd`), *ii)* the operation within the round (`rndOp`) of the cryptosystem from which the intermediate value has to be extracted, *iii)* the switching activity of the signals (`SigStat`), i.e., nTr traces of $nSampl$ samples each, *iv)* the plaintext (`ptx`) corresponding to the switching activity traces, *v)* the secret key (`key`), and *vi)* the α value used in the t-test. The `rnd` and the `rndOp` inputs are used to generate the intermediate value of the computation that is used to partition the statistics in two sets. For each bit of the intermediate value, a partition of the traces is created and the t-test analysis is executed for each collected temporal sample (see lines 12-15 in Algorithm 3). The results of the t-test are used to highlight the dependency between each bit of the intermediate value and the switching activity of the signal (see lines 14 in Algorithm 3). The algorithm returns an object that highlights, for each bit of the intermediate, if there

Algorithm 3 Specific t-test algorithm to check the vulnerability of a signal. For each bit of the intermediate, the specific t-test is executed on each time sample of the traces.

```
1: function specTtest(rnd,rndOp,SigStat[nTr, nSamp], ptx, key, α)
2:     isSigVuln = zeros(size(SigStat.nTr), 1)
3:     [tmpVal] = CalcIntermediate(rnd, rndOp, ptx, key)
4:     for iterBit = 1 : size(tmpVal) do
5:         for iterTrace = 1 : size(tmpVal) do
6:             if tmpVal[iterBit] == 0 then
7:                 Set0.add(SigStats(iterTrace))
8:             else
9:                 Set1.add(SigStats(iterTrace))
10:        for iterSample = 1 : size(SigStats.nSamp) do
11:            isVuln = ttest(Set0, Set1, α)
12:            isSigVuln[iterBit] = isSigVuln[iterBit] ∨ isVuln
13:    return isSigVuln
```

Algorithm 4 Non-specific t-test algorithm to check the vulnerability of a signal. The non-specific t-test is executed on each time sample of the traces.

```
1: function nonSpecTtest(SigStat[nTr, nSamp], α)
2:     isSigVuln = False
3:     rndTraceId = randRange(1, size(SigStats.nTr))
4:     for iterTrace = 1 : size(SigStats.nTr − 1) do
5:         Set0.add(SigStats(rndTraceId))
6:     for iterTrace = 1 : size(SigStats.nTr) do
7:         if iterTrace ≠ rndTraceId then
8:             Set1.add(SigStats(iterTrace))
9:     for iterSample = 1 : size(SigStats.nSamp) do
10:        isVuln = ttest(Set0, Set1, α)
11:        isSigVuln = isSigVuln|isVuln
12:    return isSigVuln
```

is statistical evidence that its value can be correlated with the switching activity of the signal under test.

The non-specific t-test is employed to assess the data dependency between the switching activity of the signal under test and the program data value being computed (see Algorithm 4). In contrast to the specific t-test that splits the collected statistics in two classes according to an intermediate value of the computation, the non-specific t-test partitions the statistics to obtain a set of traces collected using the same input values from one side, and a set of traces collected using random input values from the other side. It assesses the possibility of distinguish the variation in the mean between a dataset collected using fixed input values for the computation and a dataset collected by varying the processed data (see lines 12-15 in Algorithm 4). The t-test analysis is executed time-wise, i.e., for each collected temporal sample (see lines 12-15 in Algorithm 3) and its results highlight if there is statistical evidence that the switching activity of the signal under test depends on the processed data. We note that the presence of a correlation reported by at least one of the two employed t-test tests, even if involving only a single bit, prevents the use of the entire signal to build the power model described in Section III-A.

C. Power monitor implementation

Starting from the mathematical formulation of the identified power model, the `PwrImpl` stage (see Figure 2) delivers

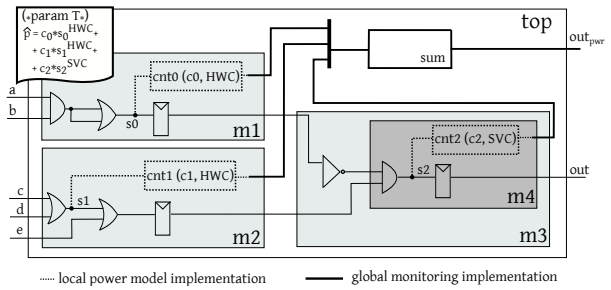


Fig. 4: Example of the power monitor architecture considering a 3-signal power model. The HWC mode is used for the signals s_0 and s_1 while the SVC mode is selected for s_2.

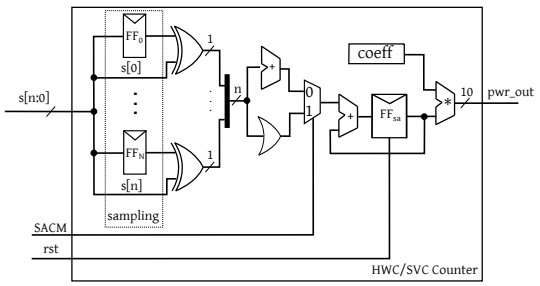


Fig. 5: Power counter architecture.

the final netlist of the target design augmented with the power monitor. In particular, the `PwrImpl` stage receives the identified power model and the *post-P&R* netlist of the target computing platform as inputs, and it produces a netlist of the computing platform augmented with the hardware description of the power monitor.

According to Equation (2), the generic *i*-th term of the power model is completely specified by a triplet of values: *i)* the name of the probed signal (s_i), *ii)* the corresponding coefficient (c_i), and *iii)* the employed Switching Activity Counting Mode (SACM), i.e., either SVC or HWC. Moreover, the temporal resolution parameter (T) specifies the time-interval between two consecutive power estimates. The realization of the identified power model encompasses two steps: local and global power monitor implementation. For each triple of values in the mathematical formulation of the identified power model, the local power monitor stage implements a customized local power counter. In particular the power counter is customized in terms of the width of the monitored signal, the coefficient associated to the signal, the Switching Activity Counting Mode (SACM) selected for the signal, and the temporal resolution (T). In contrast, the global power monitor stage implements a single power adder in the top module of the target design to deliver the power estimates. The power adder implements the mathematical structure of Equation (2). In particular, the inputs of the power adder module are connected to the outputs of each implemented power counter. For the sake of clarity, Figure 4 depicts the power monitor architecture for a 3-signal power model. The target computing platform consists of a top module (`top`) that implements three sub-modules, i.e., `m1`, `m2` and `m3`. The power monitor must consider 3 signals of the computing platform, i.e., s_0 and s_1 probed using the HWC counter type, and `s2` probed using the SVC counter type. For all the power counters, the associated coefficients (`c0`, `c1`, and `c2`) and the `SACM` are specified in the identified power model. Finally, the power adder is used to deliver the periodic power estimate $\hat{p}_t$ (see out_{pwr} in Figure 4).

We note that the power monitor implementation sits on a configurable power counter architecture (see Figure 5). The power counter periodically outputs the power contribution originated by the monitored signal as the product between the switching activity stored in the FF_{sa} memory element and the corresponding coefficient c_i assigned by the power model identification algorithm (see `coeff` in Figure 5). At the end of each power sample of period T, the signal rst resets the accumulated switching activity stored into the FF_{sa} memory element to start collecting the switching activity of the new time period. The size of the power counter depends on: *i)* the width of the monitored signal in the design, and *ii)* the maximum number of clock cycles between two consecutive power samples, i.e., the temporal resolution T. In particular, our power counter samples the monitored signal once per clock cycle to measure the switching activity in terms of the signal variations in two consecutive sampled values (see `Sampling` block in Figure 5). To provide a strong upper bound to the number of switches for each single-bit signal within a single clock cycle, our monitoring strategy avoids the direct measure of the switching activity due to glitches. However, the power consumption due to the glitches is still captured, since a higher power consumption will be associated to each toggle of the signal, i.e., the one of the actual signal toggle plus the power due to the related glitching activity, if any. We note that each power counter instance allows a design-time customization to select the way it measures the switching activity of the probed signal. At design-time, the specific Switching Activity Counting Mode (`SACM`) control input is used to configure the power counter to measure the switching activity of a signal in the form of either the Hamming Weight Count (`HWC`) or the Single Variation Count (`SVC`). For each clock cycle, the `HWC` counting mode measures the switching activity as the number of bits of the signals that flipped with respect to the previous sampled value. In contrast, the `SVC` counting mode measures the switching activity, as 0 or 1, depending if the current signal value is changed or not with respect to the previously sampled value. We note that the design-time selection of the `SACM` control signal automatically prevents the implementation of the digital logic corresponding to the unused switching activity counting mode during the implementation of the power monitor.

IV. Experimental Evaluation

This section discusses the experimental results related to the proposed power monitor design methodology in terms of accuracy loss, security against side-channel attacks as well as area and power overheads. All the considered quality metrics are measured considering the unprotected and the protected power monitors implemented into a general-purpose System-on-Chip. It has been considered the execution of different software-

implemented cryptosystems and the use of a hardware accelerator implementing the AES-128 cryptosystem. The final goal is to demonstrate the possibility of successfully performing a side-channel attack against the unprotected power monitors as well as the side-channel resistance of the protected ones, regardless the complexity of the target computing platform.

The rest of this section is organized in three parts. Section IV-A introduces the experimental setup accounting for the employed security testbed along with the formal definition of the quality metrics. Section IV-B discusses the accuracy of the power estimates and the area and the power overheads of the implemented power monitors. As our representative use cases, Section IV-C presents the security assessment results obtained by means of the two computing platforms executing different cryptosystems.

A. Experimental setup

To demonstrate the effectiveness and the generality of our solution, we implemented a power monitor targeting a general-purpose and an application-specific computing platform. As the reference general-purpose computing platform, we employ an embedded RISC-V System-on-Chip (SoC) [24]. The SoC features a 32-bit bus-based architecture and an in-order, five-stage RISC-V CPU employing the Harvard memory architecture. The CPU offers the hardware support to single-precision floating point as well as to integer divisions and multiplications. A UART peripheral and a SoC debugger complete the SoC architecture. Concerning the security assessment of the protected power monitor for the general-purpose computing platform, we employed 5 standard cryptosystems [25], [26], i.e., the 128-bit software version of the Advanced Encryption Standard (AES) [27], Clefia [28], Seed [29], Cast [30], and Camellia [31], operating in Electronic Code Block (ECB) mode. For each cryptosystem, the `C` source code has been compiled with the `LLVM` compiler framework targeting the RISC-V ISA and by using the `-O3` option. In contrast, the hardware accelerator implementing the AES-128 cryptosystem is employed as the reference application-specific computing platform [32]. The accelerator takes five clock cycles to perform each round of the AES. For each clock cycle, the implemented `SUBBYTES` primitive operates on four bytes of the AES state, thus completing the `SUBBYTES` computation in four clock cyles. The remaining round primitives, i.e., `SHIFTROWS`, `MIXCOLUMNS`, and `ADDROUNDKEY`, are computed in a single clock cycle.

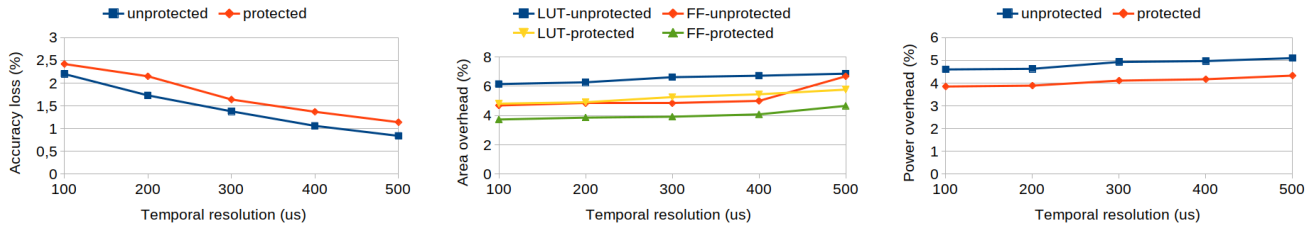
Computing platforms and power monitor design - Vivado 2018.2 has been adopted to generate the post-implementation netlist of the reference computing platform operating at 50MHz. The switching activity and the power traces to identify the power model have been collected from the post-P&R simulations with a time resolution of 200ns, i.e., one sample every 10 clock cycles. To identify the power model for the general-purpose SoC, we used the WCET benchmarks (25 applications) to generate a representative set of statistics, i.e., switching activity and corresponding power traces. We randomly selected 20 applications to identify the power model and the remaining 5 benchmarks have been used to assess the accuracy of the power estimates. Statistics have been collected by executing each application 30 times and by using a different input set at each iteration. In addition, we performed 10-thousand encryptions making use of the software version of the considered cryptosystems where each of them was fed them with a random set of plaintext and secret key values. The generated switching activity has been used to filter out the vulnerable signals during the vulnerability analysis stage. Concerning the application-specific computing platform, i.e., the hardware accelerator implementing the AES-128 algorithm, we executed a set of 10-thousand AES encryptions by using a random set of plaintext and secret key values to extract a representative set of statistics, useful to identify the power model and the vulnerable signals of such hardware accelerator.

Starting from the collected switching activity, the netlist, and the generated power traces, we employed Matlab-2019a to implement the power model identification stage and the side-channel vulnerability analysis stage. For each one of the considered computing platforms, we identified the protected and the unprotected power models. The protected power model is resistant to side-channel attacks, while the unprotected one has been identified using the same methodology without activating the side-channel vulnerability analysis stage. The latter model represents the state-of-the-art solutions, for which no vulnerability analysis is performed to filter out the signals employed in the power model. We note that the protected power models for the SoC (`protected-SoC`) and the hardware accelerator (`protected-AccHW`) are used to assess the validity of the proposed design methodology, while the unprotected ones, i.e., `unprotected-SoC` and `unprotected-AccHW`, are used to highlight the security vulnerability that affects the state-of-the-art power modeling design solutions.

To identify the vulnerable signals in the considered computing platforms, the vulnerability analysis employed both the non-specific and the specific statistical t-test tests at the granularity of the single bit, with $\alpha = 0.005$. In particular, we targeted the first exclusive-OR between the plaintext and the secret key (`ark`) and the first access to the SBOX (`sbox`) operations as the two intermediate values for the specific t-test.

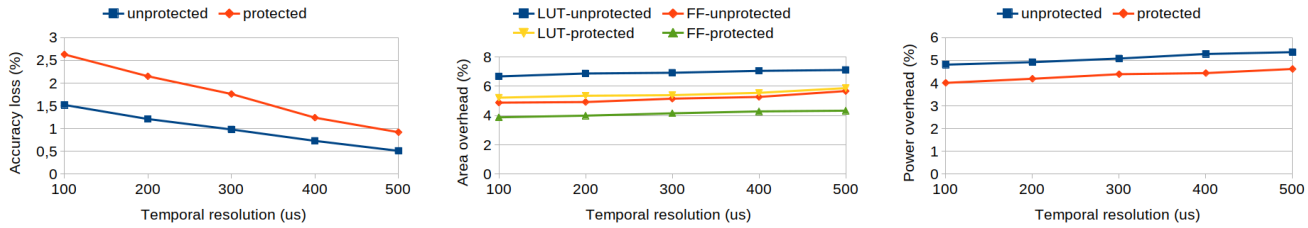
The hardware description implementing the four identified power models has been finally added to the computing platforms. In particular, Vivado 2018.2 has been employed to implement the final SoC and hardware accelerator augmented with the corresponding power monitors targeting an operating frequency fixed at 50MHz. The functional validation has been carried out by prototyping the four designs on the Digilent Nexys4-DDR board featuring a Xilinx Artix7 XC7A100TCSG324-1 FPGA. It is worth noticing that the operating frequency of the power monitor exceeds 100MHz, but we constrained the operating frequency to the one used for the System-on-Chip that shows the longest critical path.

Quality metrics: accuracy, area and power overheads - To assess the accuracy of the proposed power monitors, we employed the Root Mean Square Error (`RMSE`) quality metric. The RMSE considers the square of the distance ($\mathbf{e_i}^2$) between



(a) SoC accuracy loss (%) (b) SoC area overhead (%) (c) SoC power overhead (%)

Fig. 6: System-on-Chip executing software cryptosystems - Accuracy loss, area and power overheads for the unprotected and the protected power monitors implemented in the reference System-on-Chip. Area and power overheads are reported with respect to the reference SoC, while the accuracy loss measures the percentage error between the power estimates and the measured power consumption. The X axis is the temporal resolution T.



(a) AES hardware accuracy loss (%) (b) AES hardware area overhead (%) (c) AES hardware power overhead (%)

Fig. 7: AES hardware accelerator - Accuracy loss, area and power overheads for the unprotected and the protected power monitors implemented in the AES hardware accelerator. Area and power overheads are reported with respect to the reference hardware accelerator, while the accuracy loss measures the percentage error between the power estimates and the measured power consumption. The X axis is the temporal resolution T.

the estimated power sample ($\hat{p}_i$) and the corresponding sample of the power consumption trace (p_i), for all the n samples (see Eq. (3)).

$$RMSE = \sqrt{\frac{\Sigma_{i=1}^{n}\left(p_i - \hat{p}_i\right)^2}{n}} = \sqrt{\frac{\Sigma_{i=1}^{n}\left(e_i\right)^2}{n}} \quad (3)$$

We note that the i-th power consumption sample (p_i) can be defined as the corresponding power estimate ($\hat{p}_i$) plus the error (e_i) (see Equation (4)). Assuming a Gaussian distribution with zero mean for the error (e_i), the RMSE represents the standard deviation (σ) over the power estimates, i.e., the prediction error of our power monitor.

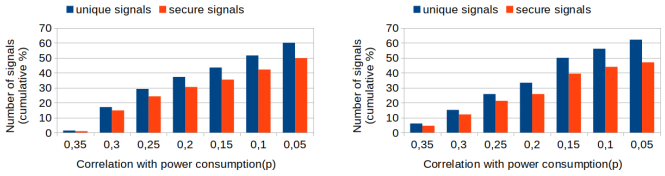
$$p_i = \hat{p}_i + e_i \quad (4)$$

To this end, we measure the final accuracy error of our power monitor by means of the normalized RMSE ($RMSE_{norm}$) which relates the RMSE value to the average $\mathbb{E}$ of the actual power consumption (see Equation (5)).

$$RMSE_{norm} = \frac{RMSE}{\mathbb{E}(p)} \times 100 \quad (5)$$

Similarly to state-of-the-art power monitoring solutions [12], [13], we assessed the accuracy of the proposed power monitors considering a set of temporal resolutions ranging from 100 us to 500 us for the two considered computing platforms. The temporal resolution measures the time distance between two consecutive power estimates. In general, a higher temporal resolution, e.g., 100us, allows to better track small and rapid variations in the power consumption, while a lower one shows a lower accuracy error due to the intrinsic averaging of the power consumption that becomes easier to be tracked. To assess the power and area overheads, we report the additional resources and power consumption due to the protected and unprotected power monitors with respect to the baseline computing platform.

Side-channel vulnerability assessment metrics - To characterize the security of the proposed solution on each computing platform, we leveraged both the correlation power analysis (CPA) and the non-specific statistical t-test performed on the power samples collected from both the unprotected and the protected power monitor implementations. The CPA analysis allows to verify the possibility of *exploiting* the side-channel information leakage in the collected trace measurements, while the statistical t-test detects the *presence* of side-channel information leakage, if any. Considering the CPA methodology, we targeted the first exclusive-OR between the plaintext and the secret key (`CPA-ark`) and the first access to the SBOX (`CPA-sbox`) operations as the two intermediate values. We note that the use of both the CPA and the t-test tool-kits are at the core of the most recently standardized procedures to assess the side-channel vulnerability of a computing platform [16].



(a) System-on-Chip (b) AES hardware accelerator

Fig. 8: Cumulative percentage of `unique` and `secure` signals of the considered computing platforms, i.e., SoC and AES-HW, available to identify the power model. Results are normalized to the total number of signals and are clustered in classes of correlation with the power consumption considering a ρ coefficient ranging from 0.35 to 0.05.

B. Accuracy loss, area and power overheads

Figure 6 and Figure 7 reports the accuracy loss ($RMSE_{norm}$) as well as the area and power overheads for the general-purpose and the application-specific computing platforms, respectively. For each computing platform, results are reported for the unprotected and for the protected power monitors, across the full range of the considered temporal resolutions.

Accuracy - To ensure an accuracy aligned to the state-of-the-art proposals [12], [13], we constrained the identification of the power models, i.e., unprotected and protected, of each computing platform to an $RMSE_{norm}$ lower than 2.7%. For each combination of computing platform and protected or unprotected power model, we implemented five power monitors, i.e., one for each temporal resolution between 100us and 500us. The accuracy loss for the protected (unprotected) power monitor of the general-purpose computing platform ranges from 2.3% (2.4%) at 100us, down to 0.87% (1.2%) at 500us (see Figure 6a). Similarity, the accuracy loss of the protected (unprotected) power monitor of the application-specific accelerator ranges from 1.5% (2.7%) at 100us, down to 0.51% (0.89%) at 500us (see Figure 7a). As expected, the accuracy error for both the computing platforms decreases with the temporal resolution since the measured power consumption signal becomes more and more regular, thus allowing the power monitor to better track it. We note that the accuracy errors of the two power monitors of the general-purpose computing platform are almost identical across different temporal resolutions and they are also aligned with the accuracy of the state-of-the-art proposals [12], [13] (see Figure 6a). In particular, the limited accuracy loss across a set of different power monitors obtained by means of different identification techniques, i.e., ours and the unprotected ones, highlights the existence of multiple accuracy-equivalent power models, where each of them makes use of a different sub-set of signals to deliver the power estimate.

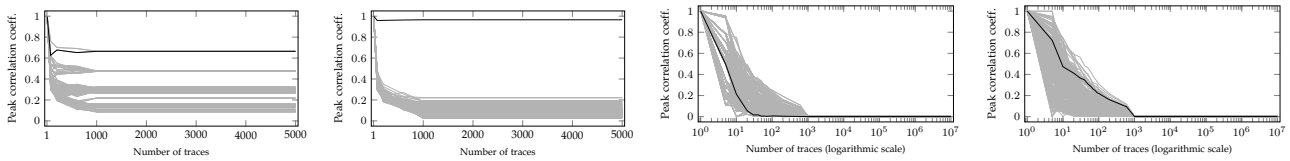
As a further investigation on this insight, Figure 8a and Figure 8b report the cumulative percentage of non-vulnerable signals for the general-purpose and the application-specific computing platforms, respectively. Such signals represent the ones that can be used to identify and implement the side-channel resistant power monitors, over the entire set of available signals of the targeted computing platform. Signals are organized in classes of correlation with the power consumption considering a ρ ranging from 0.35 (maximum correlation) to 0.05 (minimum correlation). For each class of correlation, we reported the cumulative percentage of `unique signals` and `secure signals`. The `unique signals` class contains the signals for which no other signal in the design shows a cross-correlation higher than 95%, i.e., signals with different names that show "identical" switching activities. An effective power monitor will use only `unique signals` for model identification and implementation. The `secure signals` are unique non-vulnerable signals, i.e., those signals that allows to identify a side-channel resistant power model.

Considering the general purpose computing platforms, the cumulative percentage of `unique signals` is 60%, thus demonstrating that 40% of the signals in the computing platform are not unique signals but, in contrast, their switching activity is identical to the one of other signals with a different name (see Figure 8a). Moreover, the side-channel vulnerability analysis highlights a 10% of vulnerable signals, thus lowering the cumulative percentage of `secure signals` to 50%. We note that the vulnerable signals, i.e., the ones leading to a side-channel information leakage, have been uniformly identified across the classes of correlation. Thus, the side-channel resistant power monitor can still be implemented by means of non-vulnerable signals with high correlation with the power consumption to deliver low accuracy errors. However, the non-negligible percentage of non-secure signals motivates the use of a systematic side-channel resistant identification approach to avoid the implementation of vulnerable power monitors. Similarly, the cumulative percentage of `unique signals` for the application-specific computing platform is 62%, while the cumulative percentage of `secure signals` is 48% (see Figure 8b). As for the general purpose computing platform, the leaking signals of the application-specific accelerator are uniformly distributed across all the correlation classes.

To summarize, our strategy can be described as a way to design a power monitor that employs a subset of signals for which the corresponding switching activity is non a function of the secret key and the plaintext, still allowing to achieve state-of-the-art accuracy errors in the power estimates. Such claim is supported by two facts. First, our unprotected and protected power monitors achieve almost the same accuracy for the two considered computing platforms. Second, the accuracy reported by our results for the protected power monitors is aligned with the one of the state-of-the-art proposals for which different identification techniques have been employed to derive the power model of the computing platforms [12], [13].

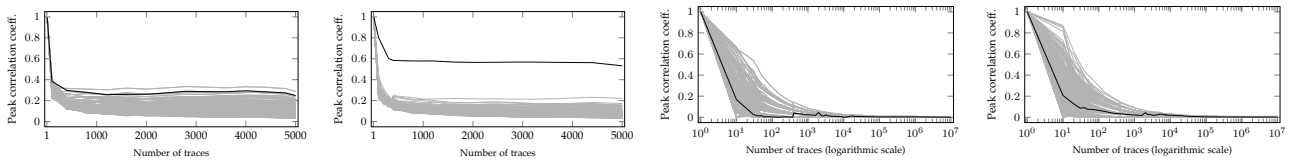
We note that our solution allows to avoid using certain signals without the need to know their actual meaning in the design, since the procedure is automatically executed. This feature is a crucial advantage since, in general, the name of a signal in a post-P&R netlist does not allow to guess its usage.

Area and power overheads - Considering different temporal resolutions ranging from 100us to 500us, Figure 6 and Figure 7 report the area and the power overheads for the power



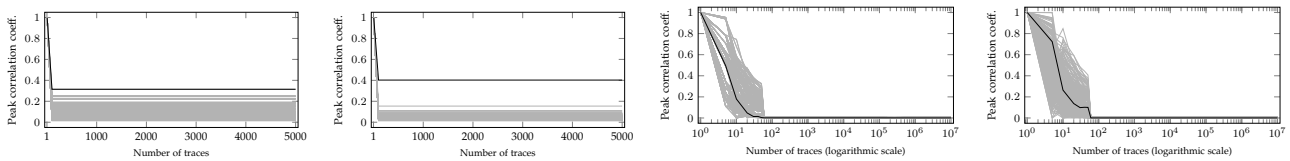
(a) `aesSW-CPA-ark` unprotect (b) `aesSW-CPA-sbox` unprotect (c) `aesSW-CPA-ark` protect (d) `aesSW-CPA-sbox` protect

Fig. 9: AES-128 (software) - Pearson's correlation coefficient results considering the execution of the software implementation of the AES-128 cryptosystem executing on the target System-on-Chip (`aesSW`). Results are reported for the `CPA-ark` and `CPA-sbox` attacks executed on the unprotected and the protected power monitor implementations. Each line depicts the evolution of the sample correlation coefficient for a key hypothesis (the correct one is highlighted in black).



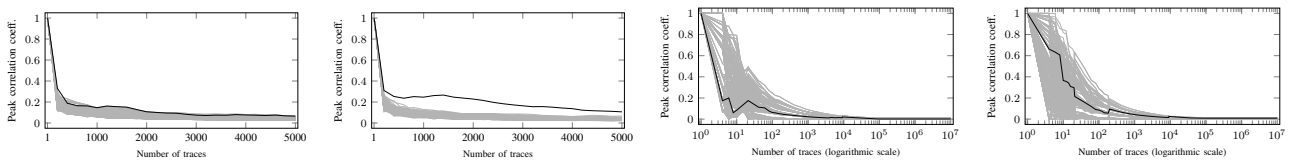
(a) `clfSW-CPA-ark` unprotect (b) `clfSW-CPA-sbox` unprotect (c) `clfSW-CPA-ark` protect (d) `clfSW-CPA-sbox` protect

Fig. 10: Clefia-128 (software) - Pearson's correlation coefficient results considering the execution of the software implementation of the CLEFIA-128 cryptosystem executing on the target System-on-Chip (`clfSW`). Results are reported for the `CPA-ark` and `CPA-sbox` attacks executed on the unprotected and the protected power monitor implementations. Each line depicts the evolution of the sample correlation coefficient for a key hypothesis (the correct one is highlighted in black).



(a) `aesHW-CPA-ark` unprotect (b) `aesHW-CPA-sbox` unprotect (c) `aesHW-CPA-ark` protect (d) `aesHW-CPA-sbox` protect

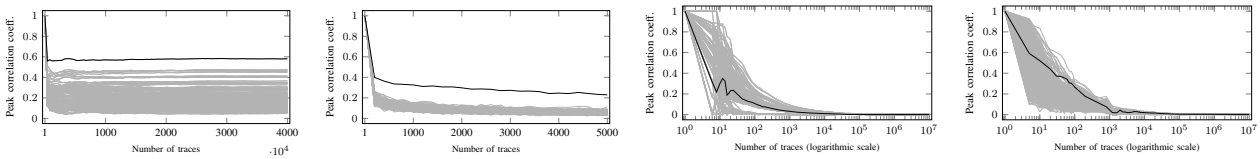
Fig. 11: AES-128 (hardware) - Pearson's correlation coefficient results considering the execution of the hardware implementation of the AES-128 cryptosystem (`aesHW`). Results are reported for the `CPA-ark` and `CPA-sbox` attacks executed on the unprotected and the protected power monitor implementations. Each line depicts the evolution of the sample correlation coefficient for a key hypothesis (the correct one is highlighted in black).



(a) `seedSW-CPA-ark` unprotect (b) `seedSW-CPA-sbox` unprotect (c) `seedSW-CPA-ark` protect (d) `seedSW-CPA-sbox` protect

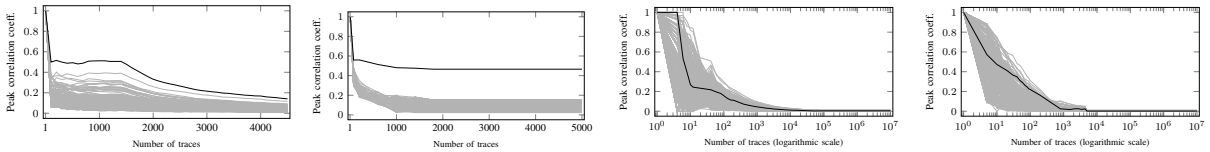
Fig. 12: Seed-128 (software) - Pearson's correlation coefficient results considering the execution of the software implementation of the Seed cryptosystem executing on the target System-on-Chip (`seedSW`). Results are reported for the `CPA-ark` and `CPA-sbox` attacks executed on the unprotected and the protected power monitor implementations. Each line depicts the evolution of the sample correlation coefficient for a key hypothesis (the correct one is highlighted in black).

monitors of the general-purpose and of the application-specific computing platforms, respectively. For each combination of power monitor, i.e., `protected` and `unprotected`, and computing platform, the area overhead is showed in terms of flip-flops (FFs) and Look-Up-Tables (LUTs). Considering the general-purpose SoC, the power overhead is lower than 5% (4% at the minimum) regardless the temporal resolution and the selected power monitor, i.e., unprotected or protected (see Figure 6c). At lower temporal resolutions, i.e., 500us, the power increase is due to the additional logic required to store and to compute the switching activity for a longer time period. We note that the difference in the power consumption between the protected and the unprotected power monitors falls into an acceptable deviation range due to the



(a) `cmllSW-CPA-ark` unprotect (b) `cmllSW-CPA-sbox` unprotect (c) `cmllSW-CPA-ark` protect (d) `cmllSW-CPA-sbox` protect

Fig. 13: Camellia-128 (software) - Pearson's correlation coefficient results considering the execution of the software implementation of the Camellia cryptosystem executing on the target System-on-Chip (`cmllSW`). Results are reported for the `CPA-ark` attack executed on the unprotected and the protected power monitor implementations. Each line depicts the evolution of the sample correlation coefficient for a key hypothesis (the correct one is highlighted in black).



(a) `castSW-CPA-ark` unprotect (b) `castSW-CPA-sbox` unprotect (c) `castSW-CPA-ark` protect (d) `castSW-CPA-sbox` protect

Fig. 14: Cast-128 (software) - Pearson's correlation coefficient results considering the execution of the software implementation of the Cast cryptosystem executing on the target System-on-Chip (`castSW`). Results are reported for the `CPA-ark` attack executed on the unprotected and the protected power monitor implementations. Each line depicts the evolution of the sample correlation coefficient for a key hypothesis (the correct one is highlighted in black).

heuristic nature of the identification algorithms. The area overheads are limited to 6% (7%) for the protected (unprotected) power monitors (see Figure 6c). As for the power overhead, the increase in the area overhead at lower temporal resolutions is due to the additional logic required to store and to compute the switching activity for a longer time period. We note that the protected (unprotected) power monitors implemented for the application-specific computing platform show an area overhead limited to 6% (7%) (see Figure 7b) and a power overhead limited to 5% (5.5%) (see Figure 7c). The overhead results for the application-specific power monitors share a similar trend with the ones used in the general-purpose SoC.

C. Security assessment

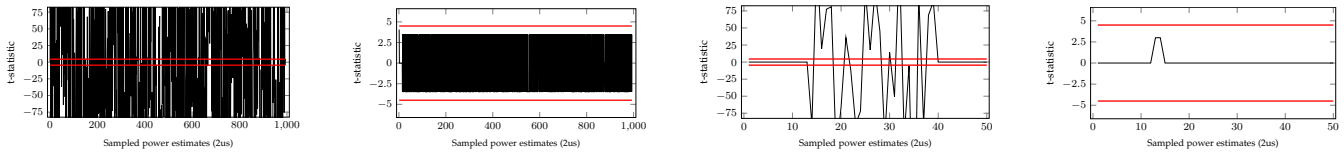
This section discusses the side-channel vulnerability of the power monitors implemented according to the proposed side-channel aware design methodology considering the general-purpose and the application-specific computing platforms. Compared to the measured power traces probed using a generic setup for emission-based side-channel attacks, the power estimate traces are almost noise-free. In fact, they are analytically computed in the power monitor starting from the values of the collected statistics and, thus, their values are not affected by the measurements and physical uncertainty. In other words, the security assessment is very conservative, since the attackers can gain access to high quality and noise-free power measurements.

Correlation power analysis (CPA) evaluation - Considering the unprotected and the protected power monitor implementations, this part discusses the experimental results of the correlation power attack against the general-purpose SoC executing AES-128 (Figure 9), CLEFIA-128 (Figure 10), Seed-128 (Figure 12), Camellia-128 (Figure 13), and Cast-128 (Figure 14). Moreover, the experimental results of the correlation power attack against the application-specific computing platform implementing the AES-128 cryptographic primitive are shown in Figure 11.

We employed the Pearson's linear correlation coefficient as the statistical tool of choice to extract the correct cryptographic key hypothesis: each line in the figures depicts the evolution of the sampled correlation coefficient for a key hypothesis (the correct one is highlighted in black), over the number of power traces considered.

For each power monitor, the CPA attack has been performed by considering the two power consumption models described in Section IV-A, i.e., `CPA-ark` and `CPA-sbox`. To provide a fair and conservative security assessment, for the unprotected power monitors the reported results correspond to the byte for which the Pearson's correlation coefficient with the correct key is smaller, i.e., the key retrieval is more challenging (see Figure 9a - 9b, Figure 10a - 10b, Figure 12a - 12b, Figure 13a - 13b, Figure 14a - 14b, and Figure 11a - 11b). In contrast, for the protected power monitors, the reported results correspond to the byte for which the Pearson's correlation coefficient with the correct key is higher (see Figure 9c - 9d, Figure 10c - 10d, Figure 12c - 12d, Figure 13c - 13d, Figure 14c - 14d, and Figure 11c - 11d).

Regardless the attack power model, the target computing platform and the executed cryptosystem, the attack against the unprotected power monitor demonstrates the possibility of retrieving the secret key with less than few hundred of power traces. In particular and according to the open literature, the `CPA-sbox` attack provides an higher Pearson's correlation coefficient value with the correct key hypothesis than the



(a) Unprotected PwrMon `SoC` (b) Protected PwrMon `SoC` (c) Unprotected PwrMon `aes-hw` (d) Protected PwrMon `aes-hw`

Fig. 15: Results of the non-specific statistical t-test test using the fixed-vs-random configuration for the unprotected and the protected power monitor implementations. Results are collected for the power monitors integrated into the general-purpose SoC executing AES-128 (Figure 15a and Figure 15b) and the power monitor implementation integrated into the AES-128 hardware accelerator (Figure 15c and Figure 15d). Note that the t-test results for Clefia executing on the general purpose SoC are not reported since their are almost identical to the ones collected for the software version of the AES.

`CPA-ark` one, thus allowing to retrieve the correct key with less than hundred power estimate traces. For example, the `CPA-ark` attack can successfully breach into the software version of the AES algorithm using 500 hundred power traces, at least (see Figure 9a), while the `CPA-sbox` requires less than 200 traces to successfully attack the same target (see Figure 9b).

In contrast, both `CPA-ark` and `CPA-sbox` attacks fail to retrieve the correct key from the protected power monitors employing up to 10 million of traces regardless the target platform and executed cryptographic primitive. For both the attacks, the Pearson's correlation coefficient values for almost all the key hypothesis falls below 0.005 after a few thousands of traces. We recall one more time that the design of the secure power monitor exploits only the switching activity of signals for which the switching activity is not a function of the secret key and the plaintext as ensured by both the specific and non-specific statistical t-test tests.

Non-specific t-test evaluation - As a further and stronger validation of the effectiveness of our secure power modeling and design methodology, we employ the non-specific statistical t-test test using the *fixed-versus-random* configuration (see Section II). The test employs a Student's t-test test considering two sets of power estimate traces extracted from two random variables with possibly different variances (also known as Welch's t-test). The test looks for a statistically significant in the difference between a set of power estimate traces collected from the execution of our testbed with a fixed plaintext in input and another set of power estimate traces collected from the execution of our testbed using a randomly varying set of plaintexts. Considering the positive security results highlighted with the CPA assessment, if the statistical test leads to accept the hypothesis that the mean values are the same (null hypothesis), we assume that the device is side-channel resistant against any non-profiled first-order attack, as changing the input does not induce differences in the behavior. According to [17], a t-statistic within the $-4.5 < t < 4.5$ interval allows to claim that the implementation is protected with a confidence $> 99.95\%$.

Figure 15 reports the results of the t-statistics from the non-specific statistical t-test executed for each collected power estimate sample -in these plots for example every 2us- produced by the unprotected (Figure 15a, Figure 15c) and the protected (Figure 15b, Figure 15d) power monitors. In particular, the t-test on the unprotected (protected) power monitors makes use of two populations of 5×10^4 (10^6) traces each. We note that the t-statistic, severely overflows the band range for the unprotected power monitors targeting the general-purpose SoC (see Figure 15a) and the application-specific accelerator (see Figure 15c). In contrast, the obtained values of the t-statistic for the protected power monitor of the general-purpose SoC (see Figure 15b) and the application-specific accelerator (see Figure 15d) lies in the security value range for the entire duration of the cryptosystem execution, thus carrying to the acceptance of the null hypothesis. We note that the t-test analysis allows to certify the absence of vulnerability for our protected power monitors, since it highlights that the power estimates of such monitors do not correlate with any computed data (including the secret key). In other words, it is impossible to setup a successful CPA regardless the employed side-channel power model.

V. Conclusions

Power monitors are widely employed to deliver online power estimates to support run-time power-performance optimization policies. However, the possibility of setting up a successful side-channel attack by analyzing such power estimates imposes the use of a secure methodology to design the power monitor. The manuscript presented a design methodology to implement accurate side-channel resistant power monitors, at hardware level, for generic computing platforms. The methodology works by designing a power monitor for which the switching activity of the signals used to compute the power estimate is not a function of the secret key and the plaintext, i.e., only non-vulnerable signals are used. By following the most recent standardized side-channel vulnerability assessment procedures [16], our experimental validation leveraged both CPA and t-test analysis. In particular, we discussed the side-channel vulnerability of different protected and unprotected power monitors tailored to both a general purpose SoC executing different standard cryptographic primitives and an application-specific accelerator implementing the AES-128 cryptosystem. The obtained results confirm the impossibility of retrieving the secret key from the power estimates of our side-channel resistant power monitor, as well as the vulnerability of the unprotected one. In addition, considering several temporal resolutions, we observed an accuracy error limited to less than

2.7%, as well as an average area and power overheads for the protected power monitors lower than 6% and 5%, respectively. In summary, the proposed methodology is able to deliver a side-channel resistant power monitor within state-of-the-art accuracy and overheads.

REFERENCES

[1] J. Szefer, "Survey of microarchitectural side and covert channels, attacks, and defenses," *Journal of Hardware and Systems Security*, vol. 3, no. 3, pp. 219–234, September 2019.

[2] D. Zoni, A. Barenghi, G. Pelosi, and W. Fornaciari, "A comprehensive side-channel information leakage analysis of an in-order risc cpu microarchitecture," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 23, no. 5, Aug. 2018.

[3] S. Bhattacharya and D. Mukhopadhyay, "Utilizing performance counters for compromising public key ciphers," *ACM Trans. Priv. Secur.*, vol. 21, no. 1, Jan. 2018. [Online]. Available: https://doi.org/10.1145/3156015

[4] O. Acıiçmez, c. K. Koç, and J.-P. Seifert, "Predicting secret keys via branch prediction," in *Proceedings of the 7th Cryptographers' Track at the RSA Conference on Topics in Cryptology*, ser. CT-RSA'07. Berlin, Heidelberg: Springer-Verlag, 2007, p. 225–242. [Online]. Available: https://doi.org/10.1007/11967668_15

[5] M. Zhao and G. E. Suh, "Fpga-based remote power side-channel attacks," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 229–244.

[6] D. Jahier Pagliari, S. Vinco, E. Macii, and M. Poncino, "Low-overhead power trace obfuscation for smart meter privacy," 06 2019, pp. 1–6.

[7] A. Barenghi, W. Fornaciari, G. Pelosi, and D. Zoni, "Scramble suit: A profile differentiation countermeasure to prevent template attacks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 9, pp. 1778–1791, 2020.

[8] R. Rodrigues, A. Annamalai, I. Koren, and S. Kundu, "A study on the use of performance counters to estimate power in microprocessors," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 60, no. 12, pp. 882–886, Dec 2013.

[9] M. J. Walker, S. Diestelhorst, A. Hansson, A. K. Das, S. Yang, B. M. Al-Hashimi, and G. V. Merrett, "Accurate and stable run-time power modeling for mobile and embedded CPUs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 1, pp. 106–119, Jan 2017.

[10] W. L. Bircher and L. K. John, "Complete system power estimation using processor performance events," *IEEE Transactions on Computers*, vol. 61, no. 4, pp. 563–577, April 2012.

[11] J. Peddersen and S. Parameswaran, "CLIPPER: Counter-based low impact processor power estimation at run-time," in *2007 Asia and South Pacific Design Automation Conference*, Jan 2007, pp. 890–895.

[12] M. Najem, P. Benoit, M. E. Ahmad, G. Sassatelli, and L. Torres, "A design-time method for building cost-effective run-time power monitoring," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 7, pp. 1153–1166, July 2017.

[13] D. J. Pagliari, V. Peluso, Y. Chen, A. Calimera, E. Macii, and M. Poncino, "All-digital embedded meters for on-line power estimation," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2018*, March 2018, pp. 743–748.

[14] D. Zoni, L. Cremona, A. Cilardo, M. Gagliardi, and W. Fornaciari, "Powertap: All-digital power meter modeling for run-time power monitoring," *MICPRO*, vol. 63, p. 12, 07 2018.

[15] NIST, "Cryptographic Module Validation Program," https://csrc.nist.gov/projects/cryptographic-module-validation-program, USA, 2020.

[16] ——, "FIPS 140 3 - Security Requirements for Cryptographic Modules," https://csrc.nist.gov/publications/detail/fips/140/3/final, USA, 2020.

[17] G. Goodwill, B. Jun, J. Jaffe, and P. Rohatgi, "A Testing Methodology for Side-Channel Resistance Validation," in *NIST Non-Invasive Attack Testing Workshop (NIAT 2011)*, 2011.

[18] George Becker, J. Cooper, Elke DeMulder, Gilbert Goodwill, Joshua Jaffe, G. Kenworthy, T. Kouzminov, A. Leiserson, M. Marson, Pankaj Rohatgi, and S. Saab, "Test Vector Leakage Assessment (TVLA) methodology in practice," in *Int. Cryptographic Module Conf., Sept. 24-26, 2013, Holiday Inn Gaithersburg, MD*, 2013.

[19] F.-X. Standaert, "How (not) to use welch's t-test in side-channel security evaluations," Cryptology ePrint Archive, Report 2017/138, 2017, https://eprint.iacr.org/2017/138.

[20] C. Whitnall and E. Oswald, "A cautionary note regarding the usage of leakage detection tests in security evaluation," Cryptology ePrint Archive, Report 2019/703, 2019, https://eprint.iacr.org/2019/703.

[21] P. C. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, "Introduction to differential power analysis," *J. Cryptographic Eng.*, vol. 1, no. 1, pp. 5–27, 2011.

[22] J. Doget, E. Prouff, M. Rivain, and F. Standaert, "Univariate side channel attacks and leakage modeling," *J. Cryptographic Engineering*, vol. 1, no. 2, pp. 123–144, 2011.

[23] L. Cremona, W. Fornaciari, and D. Zoni, "Automatic identification and hardware implementation of a resource-constrained power model for embedded systems," *Sustainable Computing: Informatics and Systems*, p. 100467, 2020.

[24] G. Scotti and D. Zoni, "A Fresh View on the Microarchitectural Design of FPGA-Based RISC CPUs in the IoT Era," *J. Low Power Electron. Appl.*, vol. 9, no. 9, p. 19, 2019.

[25] ISO/IEC 29192-2:2019, "Information security — Lightweight cryptography — Part 2: Block ciphers," USA, 2020.

[26] ISO/IEC 18033-3:2010, "Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers," USA, 2020.

[27] F. P. Miller, A. F. Vandome, and J. McBrewster, *Advanced Encryption Standard*. Alpha Press, 2009.

[28] T. Shirai, K. Shibutani, T. Akishita, S. Moriai, and T. Iwata, "The 128-bit blockcipher clefia," in *Proceedings of the 14th International Conference on Fast Software Encryption*, ser. FSE'07. Berlin, Heidelberg: Springer-Verlag, 2007, p. 181–195.

[29] H. Lee, S. Lee, J. Yoon, D. Cheon, and J. Lee, "RFC4269: The SEED Encryption Algorithm," USA, 2005.

[30] C. Adams, "RFC2144: The CAST-128 Encryption Algorithm," USA, 1997.

[31] M. Matsui, J. Nakajima, and S. Moriai, "RFC3713: A Description of the Camellia Encryption Algorithm," USA, 2004.

[32] Joachim Strömbergson, "SecWork: Verilog Implementation of the Symmetric Block Cipher AES (Advanced Encryption Standard as specified in NIST FIPS 197," https://github.com/secworks/aes.

Davide Zoni is Assistant Professor at Politecnico di Milano, Italy. He published more than 50 papers in journals and conference proceedings. His research interests include RTL design and optimization of complex embedded systems with emphasis on low power methodologies and hardware security. He filed two patents on cyber-security and he won the Switch2Product competition in 2019. He is also the principal investigator of LAMP (see www.lamp-platform.org).

Luca Cremona received his B.S. in Engineering of Computer Systems in 2014 and his M.S. in Computer Science and Engineering in 2017, both from Politecnico di Milano. He is currently a PhD student in the same university and his research interests include RTL power modeling and management for multi-cores.

William Fornaciari is Associate Professor at Politecnico di Milano, Italy. He published 6 books and around 300 papers in int. journals and conferences, collecting 5 best paper awards and one certification of appreciation from IEEE. He holds three international patents on low power design and he filed two patents on cyber-security. His research interests include embedded and cyber-physical systems ranging from sw and hw design to run-time optimizations and resource management.