Analysis and countermeasures to side-channel attacks: a hardware design perspective

Davide Zoni
Politecnico di Milano - DEIB
Via Ponzio 34/5, 20133, Milano, Italy
davide.zoni@polimi.it

***Abstract*—With the Internet-of-Things revolution, the security assessment against implementation attacks has become a critical concern not only for hardware accelerators but also for embedded CPUs executing cryptographic primitives. Starting from an FPGA implementation of the open-hardware ORPSoC system-on-chip running a software version of the AES, this paper explores two implications of the side-channel information leakage that impact the hardware design of general purpose computing platforms. First, we explore the mapping between the side-channel information leakage and the microarchitectural components that are contributing to it. Second, we discuss the variations in the observability of the side-channel information leakage at post-synthesis and post-implementation levels of the FPGA hardware design flow.**

***Index Terms*—Embedded Systems Security, Side Channel Attacks, Hardware Design Flow, Applied Cryptography**

I. Introduction

Side-channel attacks represent one of the most significant threats to the security of embedded systems in the Internet-of-Things (IoT). Traditionally, cryptographic primitives and protocols have proven to be effective and efficient means to ensure security features, i.e., confidentiality and data/endpoint authentication. However, their use in current embedded scenarios calls for a security-oriented design that takes into account both *i)* their mathematical formulation, and *ii)* their resistance against attacks led by someone having physical access to the computing platform. The latter class of attacks, known as Side-Channel Attacks (SCAs), leverages on the additional information coming from the measurement of the activity of the computing platform to infer information on the data being processed. In this scenario, power consumption represents the most exploited parameter since it has been proven to be a rich source of information and since it can be easily measured [1]. We note that a side effect of such exploitation can require to revise the traditional low power runtime strategies for IoT devices [2] that are currently neglicting security aspects.

The power analysis is the form of side-channel attack in which the attacker studies the power consumption of a cryptographic primitive, that is either software- or hardware-implemented, to retrieve the secret key. In particular, the open literature classifies the techniques exploiting the side-channel information leakage on the power consumption in two sets: simple power analysis (SPA) and differential power analysis (DPA). The techniques in the first set leverage the changes in the power consumption caused by key-dependant variations in the control flow of the computation. In contrast, the techniques in the second set leverage the changes in the power consumption caused by the different switching activity of the computing device and which are induced by the processing of different data values. In the rest of this work we focus on DPA attacks since are those for which the way the microarchitecture of the computing device is designed plays a critical role. In fact, successful SPA attacks leverage the flaws in the control flow of the algorithm, while DPA attacks are tightly coupled with the way the data is processed. To this extent, counteracting DPA imposes to break the link between the power consumption and the data being processed. Ad-hoc technology libraries [3] as well as the use of masking schemes [4] have shown to be effective in tackling DPAs. However, the additional fabrication effort induced by the former and the area and power overheads of the latter make them viable solutions to protect small cryptographic accelerators rather than general purpose embedded CPUs. To this extent, software-level coutermeasures to tackle DPAs have been proposed. These work by either randomizing the data being processed or by continuously changing the code employed to perform the computation [5]. In particular, the software architect usually takes into account the sole architectural abstraction of the computing platform, that, however, can be proven insufficient to prevent side-channel information leakage, since those latter are intimately connected to the actual implementation of the computing device, i.e., the microarchitecture.

Contributions - This work discusses the implications of the side-channel attacks in the design of general-purpose computing platforms with two contributions to the state-of-the-art. First, we explore the mapping between the observed side-channel information leakage and the microarchitectural components in the CPU that are contributing to it. Second, we motivate the variations in the observed side-channel information leakage as a function of the abstraction level at which the analysis is carried out, i.e., either post-synthesis or post-implementation.

Outline - The rest of this manuscript is organized in three parts. Section II presents the background on Differential Power Analysis. The side-channel information leakage analysis is discussed in Section III, while conclusions are drawn in Section IV.

II. Background

Given a symmetric cryptographic primitive, the differential power analysis (DPA) workflow is an instance of a known plaintext or a known ciphertext attack aiming at retrieving the secret key. The attacker is assumed to know all the implementation details of the cryptographic primitive and can measure the power consumption of the device. We note that the SCAs allow to independently consider the effect of each bit of the secret key on the power consumption of the device, thus severly reducing the security margin. Such property allows to attack small portions of the secret key and then assemble each identified portion to retrieve the entire secret key. Moreover, power-aware DPAs are effective even in the presence of a mathematically secure implementation of the target cryptographic primitive, since they leverage on the link between the power consumption and the data being processed. The DPA attack workflow 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, i.e., 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 according to the chosen leakage model [6]. This way, she can construct a hypothesized power consumption for each of the values of the attacked portion of the secret key. Last, 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 will fit best the measurements.

Leakage modeling - It is critical to decide which target intermediate value should be predicted and which data-dependant power model to use. Usually such decisions are taken on a trial-and-error basis. However, two considerations are usually of help to minimize the design space of the two above-mentioned design choices. First, the most significant portion of data dependant power consumption in CMOS devices depends on the switching power consumption of the microarchitecture. Second, a reasonable coarse model of the power consumption of the target operation producing the intermediate value is sufficient to successfully attack the target [6]. To this extent, it is common to assume that *i)* all logic components of the same kind consume the same amont of power and *ii)* the switching power remains proportional to the switching activity. As a consequence, the Hamming Distance (HD) and the Hamming Weight (HW) represent the two most popular leakage models for data dependant power consumption CMOS elements. Considering a signal in a CMOS circuit, the HD measures the number of single-bit switches between two values held by the signal in two consecutive clock cycles. Differently, the HW measures the number of ones of a signal in a specific clock cycle as the power to load its fan out. We note that the HW leakage model is simpler than the HD one since it requires the knowledge of a single value carried on the signal.

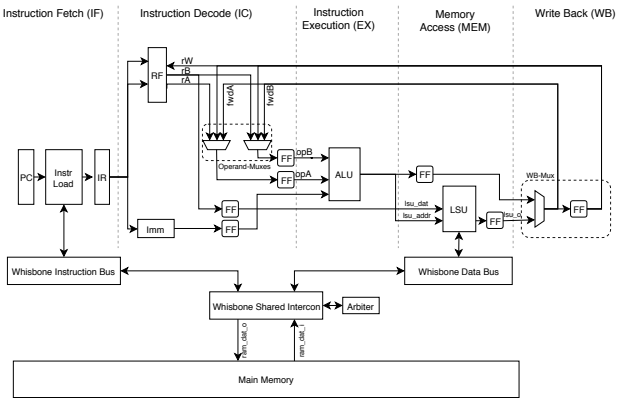


Fig. 1. Architectural view of the ORPSoC system-on-chip used for the side-channel information leakage analysis.

Statistical distinguishers - Starting from the first introduction of the Difference-on-Means (DoMs) statistical test to retrieve the secret key of a DES cipher, several other techniques have been proposed to enhance the accuracy and efficiency, i.e., reduce the number of required power traces to setup the attack, of DPAs [7]. This work considers the Welch's t-test proposed in [7]. It is an extension of the student t-test, and it represents the core statistical technique employed to check for statistical differences between two subsets of power traces of unequal sample size and variance. Equation (1) defines the t-statistic of the Welch's t-test, where μ_1, μ_2 are the sample means, n_1, n_2 the sample sizes and σ_1, σ_2 the sample standard deviations of the two populations.

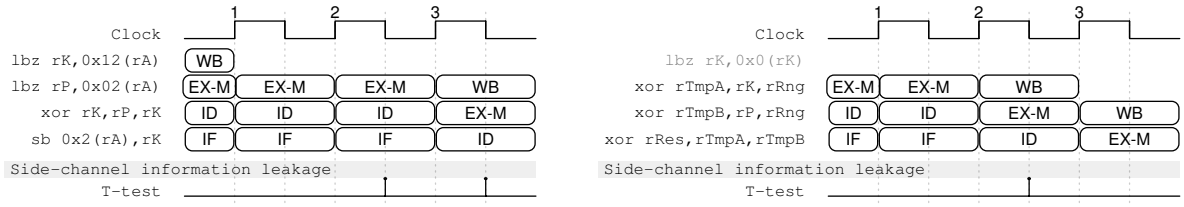
$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{{\sigma_1}^2}{n_1} + \frac{{\sigma_2}^2}{n_2}}} \tag{1}$$

Given the number of degree of freedom and a confidence level, it is possible to define, through the use of statistical tables, a threshold for the t value for which the number of degree of freedom, d, is defined in Equation (2).

$$d = \frac{(\frac{{\sigma_1}^2}{n_1} + \frac{{\sigma_2}^2}{n_2})^2}{(\frac{{\sigma_1}^4}{{n_1}^2(n_1-1)} + \frac{{\sigma_2}^4}{{n_2}^2(n_2-1)})} \tag{2}$$

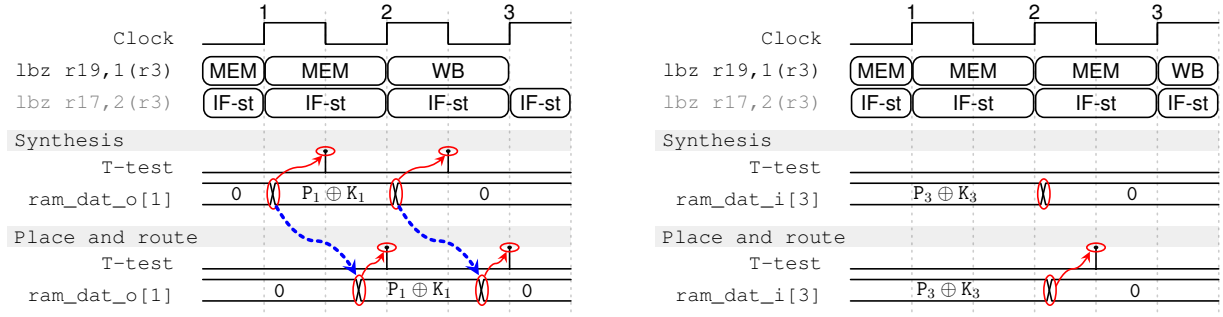
III. Side-channel information leakage analysis

This section discusses the impact of the side-channel information leakage on the hardware design flow. We consider the OpenRISC System-on-Chip (ORPSoC) Platform [8] as our reference use-case. The ORPSoC features a single-issue, in-order OpenRISC 1000 CPU with a 5 stages pipeline, and a main memory module connected to it via a Wishbone compliant bus (see Figure 1). Considering a target clock frequency of 50MHz, the complete SoC was synthesized and mapped onto a Xilinx Artix 7 XC7A100 device employing the Vivado 2017.4 toolchain, while the switching activity outputs were obtained employing Xilinx XSim 2017.4. We employed, as our case study, the Advanced Encryption Standard (AES) symmetric block cipher. In particular, we used its variant with



(a) Benchmark-1 (b) Benchmark-2

Fig. 2. Two representative examples of the side-channel information leakage due to the data serialization in the microarchitecture. Even the Boolean masking scheme can produce leakage if it serializes the secret key and the plaintext on the same input of the ALU to compute the bit-wise eXlusive OR (*xor*). *rK* and *rP* are the register storing byte of the secret key and of the plaintext, respectively. *rTmpA* and *rTmpB* registers are used to store the intermediate computation of the Boolean masking scheme, while the *rRng* contains a random value. Last, *rA* contains the base memory address of the plaintext and the secret key.



(a) Time shift of the observed side-channel leakage (b) Zero variance (synthesis)

Fig. 3. Side-channel information leakage observability at post-synthesis and post-implementation levels. The additional timing information of the post-implementation analysis can operate a time-shift to the observed side-channel information leakage (Figure 3a). Moreover, the precision of the tools can limit the observability of the security vulnerabilities in some scenarios (Figure 3b). The same sequence of two loads is analyzed targeting either the first (Figure 3a) or the third (Figure 3b) bit of the intermediate.

a 128 bit key, implemented in software, employing a standard-compliant, memory optimized (S-Box) implementation. We computed the power traces corresponding to 700 AES executions on independent, uniformly drawn, random plaintexts and the same fixed secret key (K). To assess the side-channel information leakage, we employed the Welch's t-test (see Section II) using the parameters suggested in [7]. We attack a 1-bit portion of the secret key at a time and we computed the value of the t statistic point-wise in time. We reject the null hypothesis if the value of the t-statistic is below 4.5. This, in turn, means that there is no detectable side-channel information leakage with this amount of measurements with a confidence of 99.99%. Our choice of the t-test is mutuated by the absence of noise in the simulated environment, which allows to employ such test without implicit assumptions on the distribution of the noise [9]. For each plaintext (P), we set up a t-test for each bit of the intermediate value generated after the computation of the first *AddRoundKey*, i.e., $P_i \oplus K_i$ where i is the position of the i-th bit in the 128-bit intermediate. It follows that each t-test is fed with two set of traces obtained by splitting the 700 collected traces on the actual value of the considered bit. For the sake of clarity, we will represent the outcome of the t-test, i.e., yes-no answer, instead of the t-statistic itself, simply highlighting whether, for a given time instant, the t-test detects potential side-channel leakage or not (see T-test in Figure 2 and Figure 3). We collected two energy measurements per clock cycle, i.e., first and second half of the clock period. This is in fact the minimum number of samples to observe any variation in the leakage detection due to the timing delays introduced in the Place-And-Route (PAR) step.

Microarchitectural components inducing side-channel information leakage - To expose the side-channel information leakage caused by microarchitectural CPU modules, we employed two different sequences of instructions to implement the *AddRoundKey* computation. We note that such instruction sequences are not limited to the AES primitive but they are present in the software implementations of several cryptographic libraries. *Benchmark-1* mimics a possible non-optimized sequence of instructions to perform the key addition. It loads 1 byte for each one of the two operands from the main memory, i.e., plaintext and secret key, it then applies the bitwise eXclusive-OR (*xor*) and stores the result back to the main memory. *Benchmark-2* implements a Boolean masked version of the key addition of *Benchmark-1*. It combines (*xor*) the secret key and the plaintext values with two copies of the same random value held in a third register. Then, the two obtained results are xor-ed and the final result is stored back to the main memory.

Figure 2 reports the results for *Benchmark-1* and *Benchmark-2* obtained by computing the t-statistic point-wise in time. In particular the evaluation of the CPU pipeline is

reported and for each clock cycle the possible vulnerabilities highlighted by the employed t-test are reported (see T-test waveform in Figure 2). Considering Figure 2a, we note that the t-test highlights a possible exploitable side-channel information leakage aligned to the negative edge of the clock signal in cycle 2 and cycle 3. Such leakage is due to the *data-serialization* effect that emerges when two critical data values, i.e., the plaintext and the secret key, are serialized in one or multiple microarchitectural data paths in two consecutive clock cycles. At the beginning of clock cycle 2, the second load terminates the M stage and the plaintext byte appears at the input of the Load Store Unit (LSU). Such value overwrites the previously loaded key byte and thus it generates side-channel information leakage. The same information leakage due to the serialization effect also affects clock cycle three. In particular, the plain byte traverses the flip-flip-based register in the write back stages, thus overwriting the key byte value stored in the previous cycle (see WB stage architecture in Figure 1).

Results in Figure 2b demonstrate that the data serialization effect can induce an exploitable side-channel leakage even in presence of a software-based side-channel countermeasures to DPA that are not considering the underlying microarchitecture (*Benchmark-2* implements the masked version of the *AddRoundKey* function). In particular, the assertion of the operands to the ALU (to perform the xor instructions) causes an exploitable side-channel information leakage at cycle 2. The leakage is due to the data serialization of the key byte (rK) and of the plain byte (rP) on the same physical input of the functional units, and thus onto the driver of such signal, in two consecutive clock cycles. Such transition shows a number of toggles equal to the Hamming Distance between the plain and the key bytes, and consequentially fitting the Hamming Weight power consumption model of the xor combination of the plain and of the key.

Observability of the side-channel leakage - The clean room analysis of the side-channel information leakage represents a standard practice to investigate the mapping between the security vulnerabilities and the parts of the microarchitecture that contribute to them. It is therefore important to take into account the effect of the model of the power consumption depending on the level of abstraction used for the analysis. Below we consider the side-channel leakage observability at post-synthesis and at post-implementation abstraction levels since they represents the points of the standard hardware design flow where performance, area, reliability [10] and timing metrics are measured. The *time-shift* effect emerges when the t-test detects a possible information leakage in both the post-synthesis and the post-implementation analyses. Such leakage is due to the same set of wires in both analyses, but the post-implementation one shows it delayed by half of a clock cycle with respect to the post-synthesis one (see Figure 3a considering the 1-st bit of the intermediate value). Such temporal shift is due to the actualization of the delay value for each signal made during the PAR stage, i.e., net delays are estimated or not accounted at all by the tool in the post-synthesis netlist. The *zero-variance (synth)* (see Figure 3b) captures those scenarios where the t-test detects possible side-channel information leakage sources in the post-implementation analysis and not in the post-synthesis one. While we cannot directly determine the cause of this leakage because of the closed source nature of the Vivado Power Report tool, we noted that the precision of the power model limits the observability of small power variations since the tool is not primarily intended for side-channel information leakage analysis. In particular, the power variations on the power traces in such time points go to zero due to possible truncation or rounding effects and thus no leakage can be detected. Moreover, we note that the alternate case is also possible, i.e., the leakage is shown in the post-synthesis analysis and it disappears in the post-implementation one.

IV. Conclusion and Future directions

This paper discusses the impact of the side-channel information leakage on the hardware design of general purpose computing platforms. We employ the OpenRISC SoC, and run a software version of the AES as representative use-case for the entire investigation. The SoC has been synthesized and implemented targeting the Xilinx Artix 7 XC7A100 FPGA device. Our investigation at both post-synthesis and post-implementation seems to show that the differences in the observed side-channel behavior depend on the accuracy of the considered netlist of the computing platform. Moreover, we demonstrate the link between the side-channel behavior and the microarchitectural components that are contributing to it, even in presence of a Boolean masking countermeasure.

References

[1] 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, pp. 57:1–57:30, Aug. 2018.

[2] D. Zoni, L. Cremona, and W. Fornaciari, "All-digital energy-constrained controller for general-purpose accelerators and cpus," *IEEE Embedded Systems Letters*, pp. 1–1, 2019.

[3] K. Tiri and I. Verbauwhede, "A digital design flow for secure integrated circuits," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 25, no. 7, pp. 1197–1208, 2006.

[4] O. Reparaz, B. Bilgin, S. Nikova, B. Gierlichs, and I. Verbauwhede, "Consolidating Masking Schemes," in *Advances in Cryptology–CRYPTO 2015–Part I*, 2015, pp. 764–783.

[5] G. Agosta, A. Barenghi, G. Pelosi, and M. Scandale, "The MEET Approach: Securing Cryptographic Embedded Software Against Side Channel Attacks," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 34, no. 8, pp. 1320–1333, 2015.

[6] 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.

[7] G. C. Becker, J. Cooper, E. DeMulder, G. Goodwill, J. Jaffe, G. Kenworthy, T. Kouzminov, A. Leiserson, M. E. Marson, P. Rohatgi, and S. Saab, "Test vector leakage assessment (TVLA) methodology in practice," in *International Cryptographic Module Conference*, vol. 1001, 2013.

[8] F. Jullien, J. Bennett, J. Bonn, J. Baxter, M. Gielda, O. Kindgren, P. Gavin, S. Macke, S. Cook, S. Kristiansson, S. Horne, and S. Wallentowitz. (2018) OpenRISC Reference Platform Soc ver. 3. [Online]. Available: https://github.com/openrisc

[9] F.-X. Standaert, "How (not) to Use Welch's T-test in Side-Channel Security Evaluations," Cryptology ePrint Archive, Report 2017/138, To appear in Proc. of CARDIS 2018, 2017.

[10] D. Zoni and W. Fornaciari, "Sensor-wise methodology to face nbti stress of noc buffers," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '13. San Jose, CA, USA: EDA Consortium, 2013, pp. 1038–1043.