A Simulated Approach to Evaluate Side-Channel Attack Countermeasures for the Advanced Encryption Standard

Luca Crocetti, Luca Baldanzi, Matteo Bertolucci, Luca Sarti, Berardino Carnevale, Luca Fanucci
Department of Information Engineering
University of Pisa, Pisa, Italy

***Abstract*—Modern networks have critical security needs and a suitable level of protection and performance is usually achieved with the use of dedicated hardware cryptographic cores. Although the Advanced Encryption Standard (AES) is considered the best approach when symmetric cryptography is required, one of its main weaknesses lies in its measurable power consumption. Side-Channel Attacks (SCAs) use this emitted power to analyse and revert the mathematical steps and extract the encryption key. Nowadays they exist several dedicated equipments and workstations for SCA weaknesses analysis and the evaluation of the related countermeasures, but they can present significant drawbacks as an high cost for the instrumentation or, in case of cheaper instrumentation, the need to underclock the physical circuit implementing the AES cipher, in order to adapt the circuit clock frequency accordingly to the power sampling rate of ADCs or oscilloscopes bandwidth. In this work we proposed a methodology for Correlation and Differential Power Analysis against hardware implementations of an AES core, relying only on a simulative approach. Our solution extracts simulated power traces from a gate-level netlist and then elaborates them using mathematical-statistical procedures. The main advantage of our solution is that it allows to emulate a real attack scenario based on emitted power analysis, without requiring any additional physical circuit or dedicated equipment for power samples acquisition, neither modifying the working conditions of the target application context (such as the circuit clock frequency). Thus our approach can be used to validate and benchmark any SCA countermeasure during an early step of the design, thereby shortening and helping the designers to find the best solution during a preliminary phase and potentially without additional costs.**

I. Introduction

The amount of data flowing over communication networks is continuously increasing and any disclosure or modification of such private data can have severe consequences. Typical applications in which the access should be limited only to authorized entities are industrial controls [1], private banking [2] and next-generation automotive networks [3]. Furthermore, in the Internet of Things (IoT) world every object will interact with each other, providing malicious attackers a wide surface with multiple entry points. Security countermeasures aim to ensure that the data is hidden to unwanted parties and unauthorized modifications of the content can be immediately identified.

A typical security solution for communications is symmetric cryptography [4]. This is based on the encryption of the data bit-stream performed on the transmission side and the decryption on the reception side. The encryption/decryption functions receive as input the data and a specific parameter called *key*. In symmetric cryptography the key is the same for encryption and decryption. Therefore the transmitter and receiver share this secret value and only the entities owning the key can understand the content of the message. A malicious entity monitoring the channel would notice only a meaningless flow of information. The most common symmetric encryption algorithm is the Advanced Encryption Standard, also known as Rijndael [5], released by the National Institute of Standards (NIST) in 2001 [6]. The AES can use 128-, 192- and 256-bit keys depending on the level of security required. The data is sub-divided into 128-bit blocks and each block is encrypted/decrypted independently.

The AES can be implemented both in software and hardware depending on the security needs and the performance required. Anyway both solutions show the drawback to be prone to power emission Side-Channel Attacks (SCA) [7]. Such kind of attacks belong to the category of the Power Analysis (PA) attacks, where the power consumption of the chip is the side-channel which leaks information about the secret data to be hidden to external unauthorized entities, i.e. the symmetric key. PA attacks are based on the idea that a chip implementing the AES (both a dedicated hardware unit and processor executing software code) emits power that is statistically related to the key used in the encryption process. This means that by repeatedly observing the power traces, the attacker could extract the key and understand the data flow. Extracted power traces are elaborated using mathematical techniques like first-order Differential Power Analysis (DPA) [8] and first-order Correlation Power Analysis (CPA) [9], which are the most diffused PA attacks because of their high feasibility in practice.

Several solutions have been designed to protect AES implementations against first-order PA. They are based on the modifications of the hardware or of the software to essentially re-shape the profile of the emitted power, without changing the result of the algorithm itself. Usually, a SCA countermeasure introduces a dummy power emission that is added to the original one, reducing the statistical relation with the key. Nonetheless, a PA attack can be performed even after countermeasures, but the number of power traces required to be observed is increased, making it infeasible or, at least, making the required attack time longer than the key exhaustion

time (i.e. the key cryptoperiod). The ratio of the number of power traces required to extract the key after and before the application of a specific SCA countermeasure is often used as benchmark factor to evaluate the effectiveness of that solution, thus to compare multiple solutions and choose the most appropriate one for the final application and its context of use. Evaluating and benchmarking SCA countermeasures can be costly, both in terms of time (for emitted power acquisition and its statistical analysis) and additional resources or equipments dedicated to this purpose. In this sense, the most efficient approach is the one which requires less time and less resources. While software implementations of AES can count on efficient methodologies such the ones presented in [10] and [11], the benchmark of hardware solutions can be much more challenging and expansive, also considering their physical implementation or the implementation of a suitable circuit prototype [12]. Therefore, a simulated approach can represent a good workaround to evaluate a countermeasure before its implementation and limit the effort in terms of time and cost.

We present an approach to characterize an SCA countermeasure using simulated data alone. The methodology simulates all the steps that a potential attacker performs, including power trace extraction and statistical elaboration.

The remainder of this paper is organized as follows: Section II describes SCAs on hardware implementations of the AES, Section III explains material and methods of our approach and illustrates an use case of it, Section IV analyses and discusses the results, and finally Section V gives the conclusions.

II. Side-Channel Attacks on Hardware Implementations of AES

The AES algorithm is based on the iteration of a sequence of mathematical steps called *round*. The steps which form the round are: *SubBytes*, *ShiftRows*, *MixColumns* and *AddRoundKey*. The number of rounds is 10, 12 and 14 for 128-, 192- and 256-bit master keys. Each round uses as input the output of the previous round and for each round a *round-key* derived from the master one (i.e. the secret key) is used. Therefore a key expansion algorithm to generate the round-key for each round is required. Figure 1 depicts the AES algorithm, highlighting the difference between the rounds. As shown, there is a preliminary AddRoundKey step, while the last round is less complex than the others, because it skips the MixColumns step.

A SCA is based on the concept that the power emitted into the external environment by an integrated circuit performing operations is related to the processed data. The same approach can be applied to the circuit implementing the AES round where the input data is the output of the previous round and the round key is derived from the encrypting/decrypting master key. Therefore, an entity monitoring the output and the emitted power of the AES core for a sufficient number of encryptions can disclose the key. An attacker who wants to carry out an SCA needs to collect several power traces relative to different plaintext encryptions.

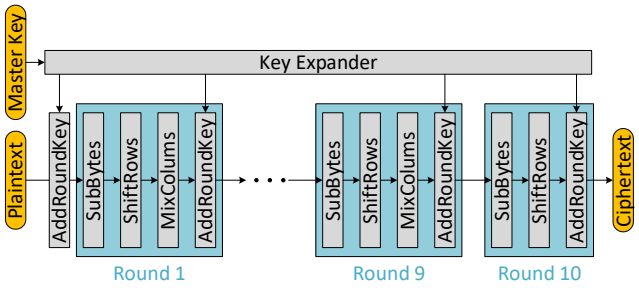

Fig. 1. The AES algorithm in the case of 128-bit keys. For 192- and 256-bit keys the number of rounds increases to 12 and 14, respectively.

Many different architectural approaches can be used when implementing the AES algorithm, but hardware solutions typically implement one round in a clock cycle. When a single round per clock cycle is executed, the implementation is called single-round pipelined, otherwise multi-round pipelined. Furthermore, in the former case the pipeline stage can be placed between each round (inter-round pipelined solution) or inside the rounds (intra-round pipelined solution). An accurate description of hardware AES implementations highlighting the difference between intra- and inter-round pipelined solutions can be found in [13]. This work focuses on single-inter-round pipelined implementations, but the approach can be extended to the other pipelined architectures with minor mathematical modifications. Figure 2 shows an example of a single-round pipelined approach in which the set of registers between different rounds can easily be identified.

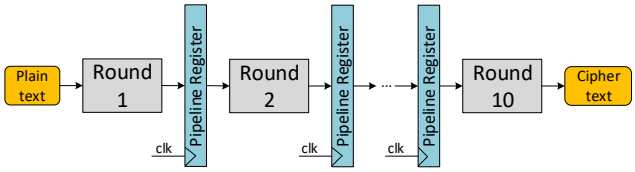

Fig. 2. The inter-round AES hardware approach. The preliminary round (preliminary *AddRoundKey* step) is included within the first round (Round 1). Other inter-round AES architectures can implement a dedicated pipeline stage for the preliminary round.

Many information about the weaknesses of the AES when implemented in hardware have been analysed in detail in several research projects [14] and some authors have highlighted the threat of power-based SCAs [15]. To solve the power emission vulnerability, various SCA countermeasures have been proposed based on modifying the AES circuit in order to change the power emission [16] and the robustness of SCA solutions has also been evaluated and characterized [17]. When attacking an AES implementation with CPA or DPA, the attack requires to target a certain element of the circuit whose logic state depends on the secret data to be extracted and a function *F()* which determines its state. Then they are acquired the power consumption traces during the

interval time in which such function is executed and finally the power consumption samples are statistically analysed basing on the respective logic state or the logic state evolution of the circuit element during such interval time. Typically the circuit element of an AES implementation attacked by CPA or DPA is the state register (i.e. the one which store the output of any round of the algorithm) or the output bus, while the function $F()$ is one of the AES rounds or a part of it. In general, assuming X as the logic state of the circuit element before the execution of $F()$ and Y as the logic state of the circuit element after the execution of $F()$, the target of the CPA/DPA attack can be expressed as:

$$Y = F(X, K) \tag{1}$$

where K is the secret key, i.e. the secret data, to be extracted and the power traces has to be acquired at least during the transition of the logic state from X to Y. Then hypotheses on the value of K are made in order to evaluate the transition $X \rightarrow Y$ and find the best correlation with the respective power trace. Basing on 1 and assuming an hypothesis K^* for K, also one between X and Y has to be known, to properly evaluate the transition $X \rightarrow Y$. Once between X or Y is known, the other logic state can be computed by applying $F()$ or its inverse $F()^{-1}$ and therefore the attack consists in:

- make an hypothesis K^* for K;
- for any power trace, compute $Y^* = F(X, K^*)$ or $X^* = F(Y, K^*)^{-1}$, assuming X or Y to be known, respectively;
- use Y^* or X^* to evaluate te transition $X \rightarrow Y^*$ or $X^* \rightarrow Y$ and perform a statistical analysis on the power traces.

For our single-inter-round pipelined architecture of the AES core, the rounds can be easily identified on the power traces because each of them corresponds to a peak of the emitted power [18]; therefore the best selection for the transformation $F()$ is one the round performed by the AES algorithm and the element of the circuit to be attacked is the pipeline register. Furthermore, the easiest round which focus on is the last one, due to its reduced mathematical complexity that lowers the computational cost of evaluating $X \rightarrow Y$. Indeed, the last AES round skips the MixColumns step that mixes data by 32-bit words at time, while the other steps of the round are byte-oriented: this allows an attacker to focus individually on each byte of the key, rather than the whole 128-bit key, therefore reducing the computational cost of the attack to the amount of only 4096 byte hypotheses (i.e. 256×16, respectively, all the possible values of a byte and all the bytes composing the 128-bit key), instead of the amount of $2^{128} \approx 3.4 \times 10^{38}$. In this case X corresponds to the last AES round 128-bit input data block and Y to the 128-bit output data block, i.e. the ciphertext, and the known data between X and Y is typically Y, referring in literature to such attack approach as the known-ciphertext attack. The guessed key K^* is the last round-key and from that the master encrypting key can be recovered by reverting the AES key expansion algorithm. Similarly, also the preliminary round can be an easy way of attacking the AES with CPA/DPA, showing the same mathematical complexity. In such case X corresponds to the plaintext and Y to the output of the first *SubBytes* operation (i.e. the *SubBytes* of the first round of AES), while the guessed key is exactly the master key; typically X is the known data and such attack scheme is referred in literature as the known-plaintext attack. Anyway, looking at a plausible attack scenario, the CPA/DPA attack on the last AES round results to be much more realistic, because the ciphertext is much more prone to be known than its respective plaintext, considering that any data transmitted on a communication bus, encrypted or not, can be sniffed, hence the ciphertext. Therefore we focus on a SCA on the last AES round. Under this assumption, Y and X become, respectively, C, the ciphertext, and I, the second to last AES round output data, K is K_{10}, the last AES round-key, and $F(I, K_{10}) = ShiftRows(SubBytes(I)) \oplus K_{10} = C$, from which

$$I^* = SubBytes(ShiftRows(C \oplus K_{10}^*)^{-1})^{-1} \tag{2}$$

Therefore the transition $I^* \rightarrow C$ is used to perform the statical analysis on the power samples and evaluate the best guess K_{10}^* for K_{10}.

Concerning the method to acquire the emitted power samples, several solutions can be employed. For instance dedicated equipments such as the DPA Workstation Analysis Platform [19] by Rambus, the ChipWhisperer toolchain [20] by NewAE Technology Inc. or the SAKURA-G/SASEBO boards [21]. Without considering the cost of the equipments, that could be very expansive in some cases, they are all solutions essentially based on FPGA platforms, thus reducing the cost for a SCA countermeasure prototype, both in terms of time and resources. Moreover they typically provide a dual FPGA platform, one for the circuit to be analysed with PA and one for all reminder parts of the design, hence isolating the SCA countermeasure and for this reason performing a more powerful analysis on it. Anyway the main disadvantage of such solutions resides in the physical implementation of the SCA countermeasure, because they require to lower the clock frequency of the circuit to some tens or units of MHz to gather the power samples, and that can be very far from the real application and attack context. Moreover such gap can be increased by the usage of FPGA platform for SCA countermeasures designed for ASIC platforms. On the other hand, in literature several authors proposed creating equivalent circuits for AES implementation to extract simulated power traces [22], anyway, very few have investigated the usage of gate-level circuits for AES models in order to guarantee a more accurate VLSI implementation model [23]. Often research focuses on oversampling when extracting power for DPA [24] and this frequently requires an extremely accurate analog model that in some contexts can be difficult to implement. A simulated approach based on the gate-level netlist of the circuit can be a good workaround, because it does not require any additional equipment and cost but the typical hardware designer EDA tools, and on the other

hand the gate-level netlist is a quite realistic approximation of the physical implementation that allows to works closer to real context conditions (such as the clock frequency). Thus we proposed a simulation environment that provides a quick and cheap solution that can be used to make validate and benchmark SCA countermeasures, without the need for any hardware prototype.

III. Material and Methods

The proposed methodology has two main steps: power extraction and statistical analysis. The following sub-sections explain the methodology by analysing in detail the power extraction and the two alternative ways of performing the statistical analysis.

A. Power extraction

The first step in our simulated SCA is based on the extraction of the power emitted by the circuit. We implemented an AES core in Verilog and then synthesized the circuit on the 45 nm CMOS technology provided by the NanGate FreePDK45 Open Cell Library. The gate-level netlist obtained as result of the synthesis represents an approximation of a real physical implementation of the same core. The switching activity of the implementation is stored in a Value Change Dump (VCD) file during the simulation of the gate-level implementation. Finally, the VCD file together with the standard-cell library leads to the power consumed for each clock cycle (i.e. each AES round). This allowed us to simulate a set of encryptions of N plaintexts and obtain N ciphertexts and a simulated power samples vector which we call $P_{simulated} = [p_0, p_1, \ldots, p_{N-1}]^T$, where every p_i element is the mean power consumed during the clock cycle in which the last AES round is performed while encrypting the *i-th* 128-bit block.

In order to extract the bytes of the key, this set of collected data needs to be processed with specific statistical approaches. We focus on the CPA and the DPA approaches, which are both based on finding a statistical relation between the $P_{simulated}$ vector and a set of expected vectors, each of which is related to a key hypothesis and is computed by evaluating the transition $I^* \rightarrow C$. The vectors with the strongest relation represent the *key guess* that matches the last round key if the attack has been successful.

Figure 3 shows the flow of the implemented power extraction in which synthesis, simulations and power extractions were performed using Synopsys Design Compiler [25], VCS [26], and PrimeTime [27], [28], respectively.

As shown in Figure 3, the extracted power can be elaborated using the DPA or the CPA analysis. Both the techniques have been implemented in MATLAB code and the following two subsections highlight the main differences between them.

B. CPA statistical analysis

The CPA is based on finding the statistical correlation between the $P_{simulated}$ vector and a set of predicted data for each byte of the key. Therefore, let us define the predicted data as a set of 16 matrices $M_0, ..., M_{15}$ where each matrix is

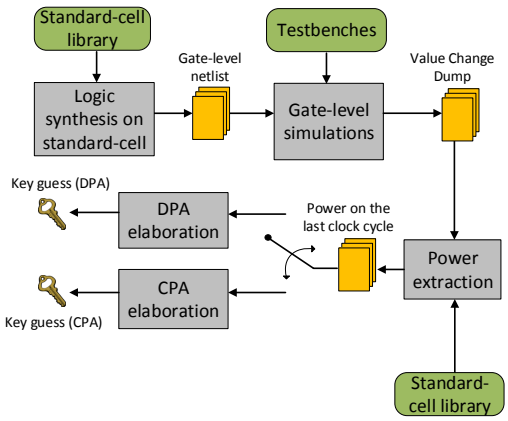

Fig. 3. The complete flow of the proposed approach.

relative to one byte b of the key. We write the matrices in the following form:

$$M_b = \begin{bmatrix} m_{b,0,0} & \cdots & m_{b,0,255} \\ \vdots & \ddots & \vdots \\ m_{b,N-1,0} & \cdots & m_{b,N-1,255} \end{bmatrix} = \begin{bmatrix} M_{b,0} \cdots M_{b,255} \end{bmatrix}$$

where $b = 0, \cdots, 15$. Each column of the matrix relates to the *h-th* key byte hypothesis, starting from 0, {00000000}, to 255, {11111111}. On the other hand, each row relates to the *i-th* ciphertext from the collection of N ciphertexts, thus for i in the range 0 to the $N - 1$.

To extract the $m_{b,i,h}$ element of the matrix, i.e. the predicted power trace for a specific byte of the key, ciphertext and key hypothesis for that byte, we use the following hypotheses, widely used in literature:

Hypothesis 1. The power consumption is proportional to the number of $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions of the pipeline register bits.

Hypothesis 2. The $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions contribute to the power consumption with the same weight.

Thus the Hamming Distance (HD) between the pipeline register state before and after the execution of the last AES round can be used to estimate the predicted power trace, as:

$$m_{b,i,h} = HD_{i,h}(C_b, I_b^*)$$

where C_b and I_b^* are the values of the *b-th* bytes of C and I^*, respectively, both for the *h-th* key hypothesis and for the *i-th* AES encryption process. Indeed, during the last round of the AES each byte of the 128-bit block is processed only by one byte of the key, and there is a unique correspondence between them. I^* is reverted from C by applying equation 2. Once the set of predicted data has been computed, the methodology simply entails finding the column in each matrix that has the highest correlation factor with the $P_{simulated}$. The correlation

factor used in our approach is the Pearson coefficient defined as follows:

$$r(x,y) = \frac{\sum_{j=0}^{n-1}(x_j - \bar{x})(y_j - \bar{y})}{\sqrt{\sum_{j=0}^{n-1}(x_j - \bar{x})^2}\sqrt{\sum_{j=0}^{n-1}(y_j - \bar{y})^2}}$$

where $\bar{x} = \frac{1}{n}\sum_{j=0}^{n-1} x_j$ and $\bar{y} = \frac{1}{n}\sum_{j=0}^{n-1} y_j$. In our case x is the $P_{simulated}$ vector and y is one column of the selected M_b matrix, hence it follows that $j = i$ and $n = N$. For any M_b matrix related to a key byte guess, we have a set of 256 Pearson coefficients $c_{b,h} = r_{b,h}(x,y)$, for $x_j = t_i$ and $y_j = m_{b,i,h}$: one Pearson coefficient for each byte hypothesis on that key byte b, i.e one $c_{b,h}$ coefficient for each column of the M_b matrix.

We call Best Guess (BG) the 16 bytes that relate to the 16 $M_{b,h}$ columns (one for each matrix) with the highest Pearson coefficient:

$$BG = BG_0 \mid BG_1 \mid ... \mid BG_{15}$$

where

$$BG_b = h^* \colon |c_{b,h^*}| = \max_{0 \le h \le 255}(|c_{b,h}|)$$

The BG represents the result of our methodology and therefore the result of our simulated attack.

C. DPA statistical analysis

We implemented a multi-bit DPA attack through the Hamming Distance model and the $m_{b,i,h}$ coefficients, instead of the typical single-bit DPA attack [10], [11]. The usage of a multi-bit approach on all the bits of byte for DPA is more effective than a single bit approach, because due to the non-linearity of the *SubBytes()* operation, or better, of the *S-box()* transformation, there is not a linear correlation between any bit of the byte and all other bits of that byte.

The proposed DPA methodology works similarly to the CPA. The $P_{simulated}$ and the set of 16 M_b matrices are computed exactly in the same way, but the BG is estimated differently. This procedure takes the $P_{simulated}$ vector and, for each column (i.e. for a specific byte hypothesis h) of a M_b matrix, splits the $P_{simulated}$ values in two groups: P_G and P_L. The selection factor to split the elements between the P_G and P_L is the following one:

$$\begin{cases} P_{G_{b,h}} \leftarrow p_i \colon m_{b,i,h} > 4 \\ P_{L_{b,h}} \leftarrow p_i \colon m_{b,i,h} < 4 \end{cases} \quad \text{for } i = 0, \cdots, N-1$$

This means that the selection function for $P_{simulated}$ is whether or not the corresponding value in the M_b column is greater or smaller than 4. Finally, the difference between the averages of $P_{G_{b,h}}$ and $P_{L_{b,h}}$ is computed:

$$D_{b,h} = |\overline{P_{G_{b,h}}} - \overline{P_{L_{b,h}}}|$$

The M_b matrix column that generates the highest "absolute difference of means" factor is the one chosen for the BG_b on byte b. Similarly to the CPA case, we can define:

$$BG_b = h^* \colon D_{b,h^*} = \max_{0 \le h \le 255}(D_{b,h})$$

The procedure is then repeated for the remaining M_b matrices, thus giving the BG for the DPA approach.

D. SCA simulation environment use case

We applied the proposed methodology to benchmark a SCA countermeasure for the AES core. Many SCA countermeasure for hardware implementations of AES can be found in literature and basically all of them aim to reduce the Signal to Noise Ratio (SNR) of the power consumption, either adding power noise or reducing the correlation between the processed data and the power emission. While the former approach typically requires the employment of Random Noise Generators (RNGs), the latter approach usually counts techniques as masking [29] or threshold implementations [30]. Among many trade-offs between area overhead, frequency decreasing and robustness level against SCA introduced, such solutions can introduce significant limitations which are higher as higher is the protection factor against SCA. Typically masking techniques can provide a high protection factor against DPA and CPA, but at the cost of a very high area overhead (usually between the 300% and the 500%, or more [29]), and of a significant clock frequency decreasing (usually between the 30% and the 55%, [29]). Threshold implementations techniques show similar features and in some cases they can also totally protect against SCA (making DPA and CPA infesable), or they can achieve better performance, in terms of area overhead and critical path overhead, but at the cost of a very lower protection factor against DPA and CPA, as shown in [30]. On the other hand solutions based on the employment of power noise generators usually do not affect at all the frequency of the AES circuit and they can offer a quite discrete robustness level against SCA at the cost of a tiny area overhead (typically between the 6% and the 11%, being higher as higher is the protection level provided). As consequence, the overall power consumption of the AES circuit significantly increases. Anyway also masking and threshold techniques require RNGs for the generations of the masks and of the shares, and typically they require more random bits than the ones required by the power noise generator approach. Thus we designed a countermeasure against DPA and CPA based on power noise generator only, because such solution can show the best efficiency in terms of ratio between the protection factor offered and the overhead factors affecting the AES circuit and because it represents however a preliminary step for the design of other countermeasures such as masking or threshold implementations, due to the need of RNGs.

Such power noise generator has been implemented through a set of several parallel Digital Ring Oscillators (DROs), that are activated when the AES core works. The DROs have two levels of enable signals: an overall enable signal, common to each ring oscillator, and a dedicated enable signal that relies

on a random bitstream. Figure 4 illustrates the high level block diagram of our SCA countermeasure.

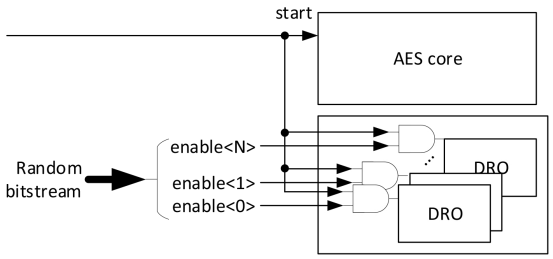

Fig. 4. High level block diagram of the DROs based SCA countermeasure against DPA and CPA.

To meet the need for random bitstreams we firstly developed a fully digital True Random Number Generator (TRNG), based on [31]. Because any simulative approach is not able to emulate the aleatory physical processes which constitute the entropy source of our TRNG and trigger the oscillators (e.g. thermal noise, temperature and voltage fluctuations), we synthesized the TRNG module on an Intel FPGA platform (EP4SGX230KF40C2) and then we gathered the required sequences of random numbers. Once the randomness of the bitstream has been tested and validated by means of the NIST Statistical Test Suite (STS) version 2, they have been used to feed the specific enable input of each DRO, during the simulations.

Anyway, also in this case some drawbacks can occur when simulating ring oscillators, because simulations cannot take in consideration the variation of the physical parameters, such as short-term and long-term fluctuations in voltage and temperature. Such physical processes randomly affect the propagation delay and transition times of the logic gates in the circuit, therefore affecting the randomness of the signals propagation along the ring oscillators and, as consequence, of their power consumption. If not integrated within the simulations, the DROs would evolve in a deterministic way, reducing the overall randomness contribution expected by such countermeasure. To emulate the uncertainty of the signals propagation delays, we introduced programmable delays among the logic gates of the DROs, reasonably of the same magnitude order of the logic gate delays variation due to temperature and voltage fluctuations. This has been realized inserting within the HDL code some delay specifications on continuous assignments and exploiting the '*define* statement of HDL preprocessor. Figure 5 shows the schematic architecture of DROs with programmable delays.

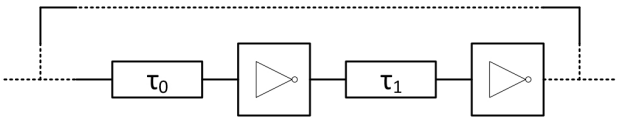

Fig. 5. DROs schematic architecture highlighting programmable delays (i.e. τ_i).

The programmable delays are statically defined at logic synthesis time and thus unmodified for the whole simulation duration. We split the overall number of AES encryption processes used for DPA and CPA in sub-blocks of 10 ones each, and for each sub-block we modified the several programmable delays, without repetitions across the sub-blocks.

IV. Results and Discussion

The proposed methodology was characterized to quantify the number of AES executions such that BG is equal to the encryption key. Figure 6 shows an execution of the technique using CPA analysis with a fixed key and a set of 8000 plaintexts. In the particular case showed, each of the 15 blue stars is a byte of the BG that correctly correspond to an encryption key byte. Figure 7 shows an execution of the technique using DPA analysis with the same data used for the CPA analysis of Figure 6: here only 12 blue stars denotes 12 bytes of the BG that match an encryption key byte. This confirms that typically CPA requires less samples than DPA to recover the whole encryption key, as expected from literature. Indeed, our methodology required around 15000 of average power samples in case of DPA and 10000 ones in case of CPA, to find a BG matching with the encryption key.

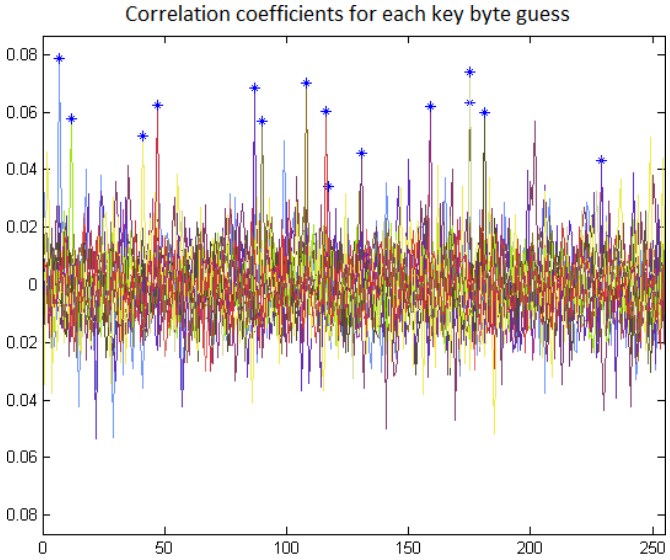

Fig. 6. Example of CPA results for 8000 samples in which each colour represents a different byte: 15 bytes of the encryption key have been recovered. On the abscissas axis the key byte hypotheses, from 0 to 255, and on the ordinates axis the corresponding Pearson coefficient values.

The number of samples required to extract the key is not a meaningful value per se, without knowing the relationship between real and simulated samples. Let's call N_s and N_r the number of samples required to extract the key from the original AES circuit in a simulated SCA and in a physical SCA, respectively.

We can hypothesize the following simplified relationship:

$$N_s = \beta N_r$$

This relationship has to be specifically characterized in the real cases, as parasitic capacitance and routing resistance heavily

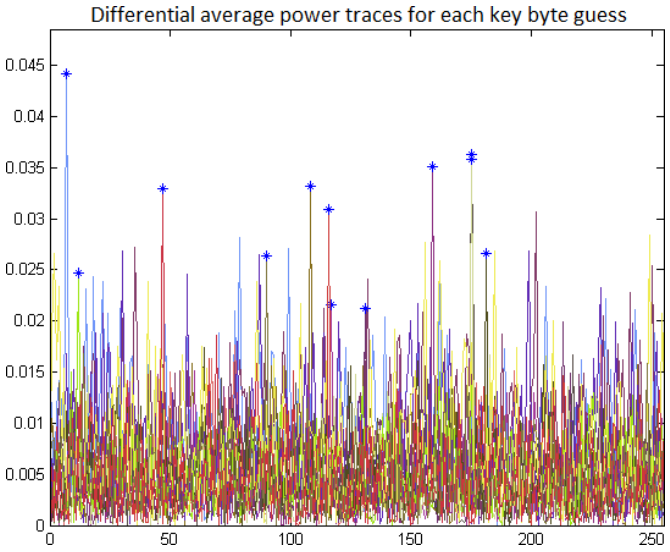

Fig. 7. Example of DPA results for 8000 samples in which each colour represents a different byte: 12 bytes of the encryption key have been recovered. On the abscissas axis the key byte hypotheses, from 0 to 255, and on the ordinates axis the corresponding "difference of averages" values.

affect power emission. The extraction and characterization of the β will be addressed in our future works.

Assuming the above expression, let's suppose that we have a set of SCA countermeasures that increase the complexity of an attack. We expect that the number of power traces that the attacker would thus have to extract in order to carry out an SCA is increased. Therefore, for each countermeasure we have $N_{s_i} = \beta N_{r_i}$. The ratio between the number of samples required to perform a real physical SCA with and without the countermeasure is the same as the ratio between the samples required to perform a simulated attack. It is thus possible to define a benchmark factor SR (i.e. SCA Resistance) that measures the effectiveness of each solution:

$$SR_i = \frac{N_{r_i}}{N_r} = \frac{\beta N_{r_i}}{\beta N_r} = \frac{N_{s_i}}{N_s}$$

The higher the SR_i value for a specific countermeasure, the higher the level of security it guarantees against SCAs. The designer can choose the best solution to be integrated into the system by comparing the SR_i parameter of different approaches and evaluating the specific trade-off of the host system.

For the proposed AES core implementation we obtained $N_s = 10^4$. Then we applied our methodology to the SCA countermeasure proposed in III-D and it required about 100 millions of average power samples for a successful CPA attack and about 150 millions one for successful DPA attack. Therefore the benchmark factor of our solution is:

$$SR_{DRO\_measure} \approx \frac{10^8}{10^4} = 10^4$$

for CPA, as well as for DPA.

Therefore we would expect to find a similar SCA resistance factor when computing the ratio between the number of average power samples required to perform a real physical SCA with and without the countermeasure proposed above.

As expected, such countermeasure does not affect at all the AES circuit critical path and the resulting area overhead is about the 7%, while the power consumption overhead is about the 300%. Table I shows the complexity in terms of gates equivalent, the maximum frequency and the average power consumption of the AES core without and with the proposed countermeasure.

Design	Area	Max. Frequency	Power consumption
AES core	17.2 kGE	580 MHz	58.6 mW
AES + proposed countermeasure	18.4 kGE	580 MHz	240.16 mW

TABLE I
FEATURES OF AES CORE AND AES CORE WITH PROPOSED SCA COUNTERMEASURE.

V. CONCLUSIONS AND FUTURE WORKS

We have described a simulated SCA for hardware implementations of the AES algorithm. When compared with other solutions [15], [19], [20] and [21], our methodology guarantees a completely simulated flow without requiring any physical circuit, additional equipment or dedicated resource to evaluate a SCA countermeasure, hence limiting the costs and saving time required for prototypes realization. A similar work was presented in [32], in which the authors showed a comparison of the robustness against SCA of the same implementation of the AES realized using two different technologies, CMOS and MCML (MOS Current Mode Logic). The authors used a transistor level simulation methodology using SPICE software while managing the coordinated simulation of sub-portions of the circuit, to reduce the high simulation time. In our work we present instead a methodology to compare and benchmark different countermeasures realized with the same technology. Also [33] presents a similar approach, in which the authors use a known-plaintext CPA attack context for the AES key guess and lower the AES circuit frequency to 2.5 MHz, in order to oversample the power consumption at the frequency of 1 GHz within the simulations. Instead our methodology implements the known-ciphertext attack, that can exhibits a wider application range than its counterpart in a real-world scenario, and it relies on the acquisition of the mean power consumed during the clock cycle, without need of oversampling, neither altering the AES circuit clock frequency and reducing the amount of collected data (thus also of the simulation time as well). Moreover, in the DPA case, we implemented a multi-bit approach which is more efficient than the typical single-bit approaches as [10], [11]. The proposed simulated SCA methodology is flexible for several reasons: it can be integrated with any EDA tools at disposal of the design team (for instance using QuestaSim [34] by Mentor Graphics in place of VCS for the simulation phase); it can be easily adapted to any AES hardware architecture (for instance to an intra-round pipelined architecture) with some minor mathematical modification, as

the target transformation $F()$ and, accordingly, to the attack point in time (i.e. the clock cycle to be analysed) and the known data to be used (e.g.: ciphertext, plaintext, s-boxes output, ...); it can provide evaluation and benchmark of SCA countermeasures for any technology (both FPGA and ASIC), according to the technologies libraries at disposal of the design team.

The SCA results would obviously be different in a real-world scenario, but our techniques could be easily used to characterize countermeasures before prototyping, reducing the time-to-market of designed physical solutions. The results are therefore valid as a benchmark for comparing SCA countermeasures and measuring their effectiveness, also being the analysis leaded in conditions much closer to the a real-world scenario than our methodologies. Our approach gives a quick result thus reducing the time required to choose the right solution for a given context and shortening the characterization phase.

Future works should include:

- a characterization of the relationship between simulated and physical results;
- the benchmarking of several other countermeasures implementations using this methodology;
- the integration of the parasitics in the simulation flow;
- the integration of additional models for the target function $F()$ and attacking points;
- the development of a GUI, by means of the MATLAB App Designer tool.

ACKNOWLEDGEMENTS

This work has been partially supported by Intel.

REFERENCES

[1] E. D. Knapp and J. T. Langill, *Industrial Network Security: Securing critical infrastructure networks for smart grid, SCADA, and other Industrial Control Systems*. Syngress, 2014.

[2] F. Ramzan and T. Pervaiz, "Online Banking Security."

[3] C. Robinson-Mallett, "Automotive Security: From Security to Safety Issues through the Introduction of In-Car Internet Connectivity," in *Invited Talk on 8th Cyber Security and Information Intelligence Research Workshop*, Jan. 2013.

[4] S. Chandra, S. Bhattacharyya, S. Paira, and S. S. Alam, "A study and analysis on symmetric cryptography," in *Science Engineering and Management Research (ICSEMR), 2014 International Conference on*, Nov. 2014, pp. 1–8.

[5] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[6] "Advanced Encryption Standard (AES)," *NIST Federal Information Processing Standards Publication (FIPS)*, vol. 197, Nov. 2001.

[7] M. Z. Rahaman and M. A. Hossain, "Side channel attack prevention for AES smart card," in *Computer and Information Technology, 2008. ICCIT 2008. 11th International Conference on*, Dec. 2008, pp. 376–380.

[8] Z. Martinasek, V. Clupek, and T. Krisztina, "General scheme of differential power analysis," in *Telecommunications and Signal Processing (TSP), 2013 36th International Conference on*, Jul. 2013, pp. 358–362.

[9] H. Pahlevanzadeh, J. Dofe, and Q. Yu, "Assessing CPA resistance of AES with different fault tolerance mechanisms," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan. 2016, pp. 661–666.

[10] E. Brier, C. Clavier, and F. Olivier, "Correlation Power Analysis with a Leakage Model," in *Cryptographic Hardware and Embedded Systems - CHES 2004*. Springer, 2004, pp. 16–29.

[11] N. Veshchikov, "SILK: High Level of Abstraction Leakage Simulator for Side Channel Analysis," in *Proceedings of the 4th Program Protection and Reverse Engineering Workshop*. ACM, 2014, pp. 3:1–3:11.

[12] W. F. Lee, "ASIC Design Flow," *Verilog Coding for Logic Synthesis*.

[13] D. Kotturi, S.-M. Yoo, and J. Blizzard, "AES crypto chip utilizing high-speed parallel pipelined architecture," in *IEEE International Symposium on Circuits and Systems 2005.*, 2005, pp. 4653–4656.

[14] N. T. Courtois and G. V. Bard, "Algebraic Cryptanalysis of the Data Encryption Standard," in *Cryptography and Coding*. Springer, 2007, pp. 152–169.

[15] S. B. Ors, F. Gurkaynak, E. Oswald, and B. Preneel, "Power-analysis attack on an ASIC AES implementation," in *Proceedings of the Information Technology Coding and Computing (ITCC)*, vol. 2, Apr. 2004, pp. 546–552.

[16] A. G. Bayrak, F. Regazzoni, D. Novo, P. Brisk, F.-X. Standaert, and P. Ienne, "Automatic Application of Power Analysis Countermeasures," *IEEE Transactions on Computers*, vol. 64, pp. 329–341, Feb. 2015.

[17] M. Masoumi, P. Habibi, and M. Jadidi, "Efficient implementation of masked AES on Side-Channel Attack Standard Evaluation Board," in *Information Society (i-Society), 2015 International Conference on*, Nov. 2015, pp. 151–156.

[18] E. Oswald, S. Mangard, N. Pramstaller, and V. Rijmen, "A side-channel analysis resistant description of the AES S-box," in *International Workshop on Fast Software Encryption*. Springer, 2005, pp. 413–423.

[19] Rambus DPA Workstation Analysis Platform. [Online]. Available: https://www.rambus.com/security/dpa-countermeasures/dpa-workstation-platform/

[20] NewAE ChipWhisperer toolchain. [Online]. Available: https://newae.com/tools/chipwhisperer/

[21] SAKURA/SASEBO board. [Online]. Available: http://satoh.cs.uec.ac.jp/SAKURA/hardware/SAKURA-G.html

[22] K. Iokibe, K. Maeshima, T. Watanabe, and Y. Toyota, "Security simulation against side-channel attacks on Advanced Encryption Standard circuits based on equivalent circuit model," in *2015 IEEE International Symposium on Electromagnetic Compatibility (EMC)*, Aug. 2015, pp. 224–229.

[23] M. Nagata, D. Fujimoto, and D. Tanaka, "Power current modeling of cryptographic VLSI circuits for analysis of side channel attacks," in *Electromagnetic Compatibility (APEMC), 2013 Asia-Pacific Symposium on*, May 2013, pp. 1–4.

[24] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Annual International Cryptology Conference*. Springer, 1999, pp. 388–397.

[25] *Synopsys Design Compiler*. [Online]. Available: https://www.synopsys.com/support/training/rtl-synthesis/design-compiler-rtl-synthesis.html

[26] Synopsys VCS. [Online]. Available: https://www.synopsys.com/verification/simulation/vcs.html

[27] Synopsys PrimeTime. [Online]. Available: https://www.synopsys.com/implementation-and-signoff/signoff/primetime.html

[28] G. Yip, "Expanding the Synopsys PrimeTime solution with power analysis," *Synopsys, Inc., http://www.synopsys.com*, 2006.

[29] N. Kamoun, L. Bossuet, and A. Ghazel, "A masked Correlated Power Noise Generator use as a second order DPA countermeasure to secure hardware AES cipher," in *ICM 2011 Proceeding*, Dec. 2011, pp. 1–5.

[30] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen, "Trade-Offs for Threshold Implementations Illustrated on AES," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, pp. 1188–1200, Jul. 2015.

[31] M. Dichtl and J. D. Golić, "High-Speed True Random Number Generation with Logic Gates Only," *Cryptographic Hardware and Embedded Systems - CHES 2007*, vol. 4727, pp. 45–62, 2007.

[32] F. Regazzoni, T. Eisenbarth, A. Poschmann, J. Großschädl, F. Gurkaynak, M. Macchetti, Z. Toprak, L. Pozzi, C. Paar, Y. Leblebici, and P. Ienne, "Evaluating Resistance of MCML Technology to Power Analysis Attacks Using a Simulation-Based Methodology," in *Transactions on Computational Science IV*. Springer, 2009, pp. 230–243.

[33] Z. Zheng, X. Zou, Z. Liu, and Y. Chen, "Security Analysis and Optimization of AES S-boxes Against CPA Attack in Wireless Sensor Network," in *2007 International Conference on Wireless Communications, Networking and Mobile Computing*, Oct. 2007, pp. 2608 – 2612.

[34] Mentor Graphics Questa Advanced Simulator. [Online]. Available: https://www.mentor.com/products/fv/questa/