Simulation-Time Security Margin Assessment against Power-Based Side Channel Attacks

Alessandro Barenghi
DEI - Dipartimento di Elettronica e Informazione
Politecnico di Milano
Via Ponzio 34/5, 20133 Milano, Italy
barenghi@elet.polimi.it

Gerardo Pelosi
DEI - Dipartimento di Elettronica e Informazione
Politecnico di Milano
Via Ponzio 34/5, 20133 Milano, Italy
pelosi@elet.polimi.it

Francesco Regazzoni
ALaRI - University of Lugano
Via Buffi 13, 6900 Lugano, Switzerland
regazzoni@alari.ch

ABSTRACT

A sound design time evaluation of the security of a digital device is a goal which has attracted a great amount of research effort lately. Common security metrics for the attack consider either the theoretical leakage of the device, or assume as a security metric the number of measurements needed in order to be able to always recover the secret key. In this work we provide a combined security metric taking into account the computational effort needed to lead the attack, in combination with the quantity of measurements to be performed, and provide a practical lower bound for the security margin which can be employed by a secure hardware designer. This paper represents a first exploration of a design-time security metric incorporating the computational effort required to lead a power-based side channel attack in the security level assessment of the device. We take into account in our metric the possible presence of masking and hiding schemes, and we assume the best measurement conditions for the attacker, thus leading to a conservative estimate of the security of the device. We provide a practical validation of our security metric through an analysis of transistor-level accurate power simulations of a 128-bit AES core implemented on a 65 nm library.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application Based Systems**]: Microprocessor/microcomputer applications; C.5.3 [**Computer System Implementation**]: Microcomputers—*portable devices*; E.3 [**Data Encryption**]: Standards (AES)

General Terms

Security

Keywords

Side-Channel Attacks, Security Metrics, Design Time Security Evaluation, Correlation Power Attack, Secure Hardware

1. INTRODUCTION

Side channel attacks based on power consumption analysis are a well known and practical threat to the security of real world implementations of cryptographic primitives. Among the possible implementations, the ones where it is harder to mitigate an effective attack after it has been discovered are surely the ASIC ones, as it is not possible to change the on-die circuit to patch a vulnerability after it has been exposed to the public. To this end, the designers of secure ICs perform extensive testing of the prototype chips in order to determine their resistance against side-channel attacks. However, performing post tape-out tests implies providing a feedback on the effective security margin of the device in the late stages of the design and production flow. It is thus desirable to have a metric, applicable at design time, providing a conservative bound for the security margin provided by an implementation, possibly expressed in terms of the computational effort required in order to overcome the protection provided by it. The idea of quantifying the computational effort required by a side-channel attack to succeed allows the designer to tune the amount of countermeasures in such a way that, even assuming the attacker is able to perform extremely precise measures of the side-channel of interest, even in ideal conditions, he is not able to effectively crack the cryptosystem. This point of view is becoming more and more of interest as cheap, off-the-shelf massively parallel architecture are becoming more and more common, providing effective platforms to perform large-computational effort cryptanalyses [1, 2, 4]. This in turn provides room to lead side-channel attacks employing a large number of key hypotheses, which were previously thought to be "very hard" in qualitative terms. To this end, this work provides a tentative metric to gauge the security margin of a cryptographic device against power analysis based side-channel attacks at simulation time, assuming conservatively an attacker able to perform perfect measurements, and in full knowledge of the circuit implementation. For this scenario we provide computational complexity bounds and a sound estimate of the number of measurements needed in order to effectively extract the secret key embedded in the secure device, given a chosen statistical confidence.

The work is organized as follows: in Section 2 we provide the background on the typical design flow employed to estimate the power consumption of a digital circuit at design time, while in Section 3 we propose our security metric, provide the bounds on complexity and evaluate the effect of employing hiding and masking countermeasures to protect cryptographic primitives implementation. In Section 4 we provide a description of the architecture we chose to employ as a case study for our metric, while Section 5 provides the security evaluation of the case studies evaluating their base resistance against side-channel attacks, and the level of protection needed to reach an 80-bit security level against side-channel attacks. Finally, Section 6 draws our conclusions.

2. POWER CONSUMPTION MODELING

Willing to provide a design time metric for the security of a chip, we will focus on simulation-time evaluation of the information leakage from a cryptographic circuit. This choice is justified by the capability to provide instrumental noise-free data regarding the power consumption of a circuit, with a timescale beyond the one measurable with current state of the art instruments, in turn implying a conservative estimation of the security margin provided by the actual implementation. To this end, we recall briefly the typical simulation flow employed to obtain power consumption estimates from a RTL specified chip design. The typical simulation flow, depicted in Figure 1 involves as inputs the RTL design of the target circuit and a technology library with which the aforementioned circuit will be realized in practice. The workflow starts by providing

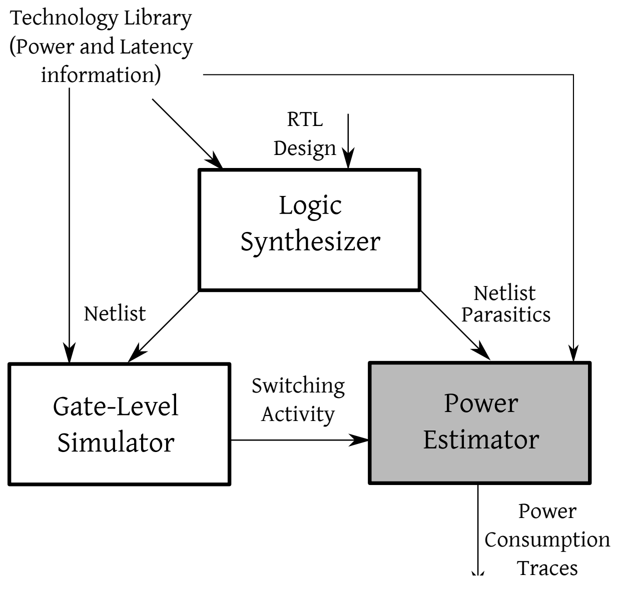


Figure 1: Typical circuit simulation workflow

a Register Transfer Level (RTL) description of the circuit to a logic synthesizer: this description is usually obtained through processing a high level description of the design through a proper synthesis tool. The RTL description is complemented by a technology library, providing the information concerning the actual logic building blocks to the logic synthesizer tool. In particular the technology library provides the available atomic parts (i.e. logic gates, latches and occasionally other elementary parts such as muxes) which can be employed when mapping the RTL description of the circuit into a placeable and routable one. The last module of the logic synthesizer will output the description of a completely placed and routed circuit, in the form of a text file commonly called netlist. This circuit respects the constraints imposed by the designer in terms of area and longest critical path and describes closely the actual silicon implementation of the digital circuit. The obtained netlist can be employed to perform a functional simulation of the circuit through a gate level simulator, which will yield a time-accurate log of the switching activity of the circuit, taking into account the signal propagation delays caused by the gates, as this information is contained within the technology library. The switching activity information output by the gate level simulation tool is fed, together with the netlist describing the circuit, and the technology library employed to obtain both, to the actual power estimation tool of the digital design toolchain. This tool, considering the input data together with the parasitic capacitances caused by the wiring of the circuit, simulates accurately the current flows in the circuit, producing as output a very accurate log of them. In particular, the class of power estimation tools we will be considering in this work is the one of transistor-level power simulators, which are effectively able to produce a simulated current log with a precision depending on the actual models of the transistors provided within the technology library. Usually, the minimum timescale for such simulations is in the picoseconds range. In particular, the currents obtained as a result of such a simulation are completely free from any measurement noise or disturbance which may be introduced by a physical measurement instrument, and thus represent a very accurate model of the circuit under exam.

3. SECURITY METRIC

The capability of building a power model approximating the actual power consumption of a cryptographic device is the first and fundamental step to lead a differential power analysis. This is due to the fact that power analysis exploits the correlation between the power employed by a device to carry out a computation and the values being elaborated. The ability of the attacker to build an accurate model of the power consumption of the targeted architecture relies on the depth of his knowledge of the implementation.

Typically, as the details may not be fully known, general models for the power consumption of vulnerable operations are the Hamming weight and the Hamming distance, respectively. Indeed, the dynamic power consumption of the aforementioned operations is directly proportional to the switching activity triggered by operations such as bitwise computations or loads/stores on registers.

By contrast, a design time simulation of a gate-level description of the targeted device provides a very accurate estimate of the power consumption profile, with a particularly fine timescale. In particular this circuit simulation is free from measurement noise and sports near-infinite bandwidth as there are no physical constraint on the precision of the measure, and it is performed directly on the on-die wire to be probed. This, in turn, implies that the circuit designer is able to simulate a conservative approximation of the most powerful attacker possible, as he is able to obtain both a higher accuracy (in terms of noise-free measurements) and precision than the usual oscilloscope-based measures adopted by an attacker. We note that, employing a transistor level power simulation, we are bounded in our precision only by the precision of the actual simulators. Through exploiting this capability, the circuit designer is able to evaluate a sound lower bound of the computational effort required by an attacker trying to breach the security of the device being designed, and consequently take appropriate actions before the device is actually manufactured.
The following definition provides a comprehensive and formalized description of this sequence of observations.

DEFINITION 3.1 (PERFECT ATTACKER). *The scenario with the most advantageous position for an attacker willing to analyze the leakage of a power-based side-channel is characterized by: (1) Knowledge of the target architecture at post-place-&-route gate level. (2) A complete characterization of the power profile of the vulnerable operations for all possible inputs. (3) Perfect Knowledge of the time instant when the vulnerable operation leaks the maximum information (i.e. the time instant when the power con-*

sumption exhibits the highest variance w.r.t. the possible inputs). (4) The absence of instrumental or environmental noise.

Note that, "template attacks" build a-posteriori models which (in the best case) fits the a-priori knowledge of the aforementioned Perfect Attacker, as they characterize the behavior of a clone of the target device varying the inputs. Consequentially, the Perfect Attacker (i.e. the designer) is able to employ the common Pearson correlation coefficient to determine which of its consumption hypotheses fits best the actual one, as the model in his possession fits the actual consumption up to a constant multiplicative factor. However, as he will estimate the actual correlation coefficient through computing its sample estimator, it is crucial that enough samples are available to him, so to effectively obtain a statistically sound estimation for a given confidence level. We will now analyze the computational complexity of conducting a correlation power analysis depending on the size of the parameters involved.

PROPOSITION 3.1. *Let k be the bit-length of the secret key to be recovered, and let w be the number of bits of the secret key that are employed by the targeted design to compute the observed vulnerable operation iteratively. Assume that the attacker is able to record n_{tr} measurements (traces), each of them l samples long, and denote as b the number of bits employed in formulating the key-hypothesis ($1 \le b \le w$) of the correlation-based attack (CPA). The computational complexity of a CPA led by a Perfect Attacker against an un-protected device is: $\mathcal{O}\big(\frac{k}{b}\, l\, \mathcal{F}(n_{tr}, b)\big)$, where $\mathcal{F}(n_{tr}, b){=}4\, n_{tr}\, 2^b$.*

PROOF. Considering a specific time instant of the cryptographic algorithm execution, let X be the random variable representing the actual measured consumption for each possible input of the observed vulnerable operation and let Y be the random variable modeling the predicted power consumption given a specific key-hypothesis. Computing the sample Pearson correlation coefficient

$$r_{XY} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\ \sqrt{n\sum y_i^2 - (\sum y_i)^2}}$$

where $\{x_1, \ldots, x_{n_{tr}}\}$ and $\{y_1, \ldots, y_{n_{tr}}\}$ are the n_{tr} sample values of the random variables X, Y, respectively, has a complexity of $\mathcal{O}(8\, n_{tr})$[1]. A straightforward approach for the attacker implies that 2^b correlation coefficients are computed independently, yielding a complexity of $\mathcal{O}(8\, n_{tr}\, 2^b)$. However, since only the Y variable changes value during all the computations, it is possible to pre-compute and store the values depending only from X thus effectively lowering the total complexity from $\mathcal{O}(8\, n_{tr}\, 2^b)$ to $\mathcal{O}(4\, n_{tr}\, 2^b)$. Taking into account the fact that this computation must be done for all the l samples of the measurement, and the whole key should be retrieved, the computational complexity of the whole CPA attack is thus $\mathcal{O}\big(l\, \frac{k}{b}\, 4\, n_{tr}\, 2^b\big)$. $\square$

The computational complexity of the CPA provides a mean to derive the optimal values of the free variables n_{tr} and b (i.e. the number of traces and the number of key hypotheses) for the attacker to perform the least possible computational effort. Nonetheless, these two variables are non-trivially bound as the number of traces depends on the values of the correlation coefficients corresponding to each key hypothesis. In particular, for a successful CPA, it should hold that the correlation coefficient for the correct key is the highest one and the confidence interval for it does not overlap with the ones of the correlation coefficients of other key guesses [3, 7, 15]. To this end, we recall the width of the bilateral confidence interval for the sample correlation coefficient r, given the number of traces n_{tr} and the error probability α.

[1]We recall that x_i represents the instantaneous power consumption of the device computing with the i-th plaintext as input at a certain time instant, while y_i is the consumption predicted for the same computation, given a fixed value of the key hypothesis.

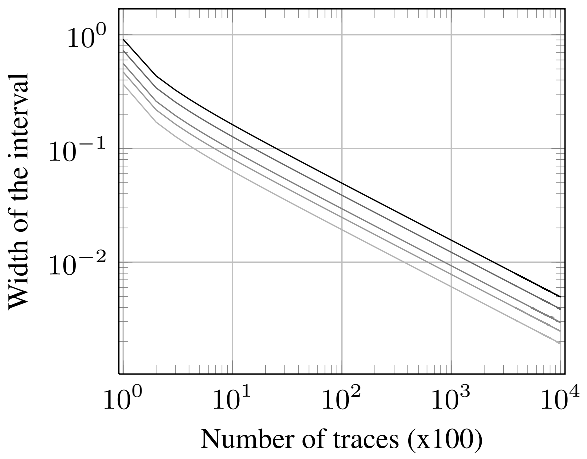


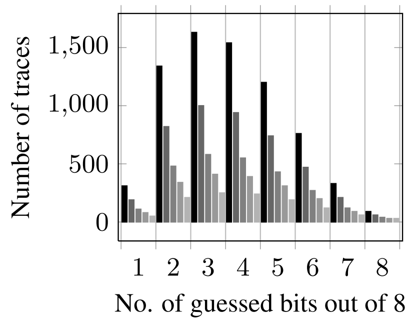
Figure 4: Width of the bilateral confidence interval, $\alpha{\in}\{0.2, 0.1, 0.05, 0.01, 0.001\}$, raising the number of traces

DEFINITION 3.2 (CONFIDENCE INTERVALS). *Given a sample correlation coefficient value r obtained with n_{tr} samples, its confidence interval $I_r{=}[\xi_{inf},\ \xi_{sup}]$ for a chosen error probability α, defines the continuous interval where the actual coefficient resides with probability $1-\alpha$. Alternatively the confidence interval can be seen as the margin of uncertainty in the estimation of the theoretical value of correlation, given a confidence of $1{-}\alpha$. The size of the confidence interval, $|I_r|{=}2\frac{z_{1-\alpha/2}}{\sqrt{n-3}}$, decreases as the chosen value for the error probability α increases, and as the number of samples n_{tr} employed in the estimation grows* [2].

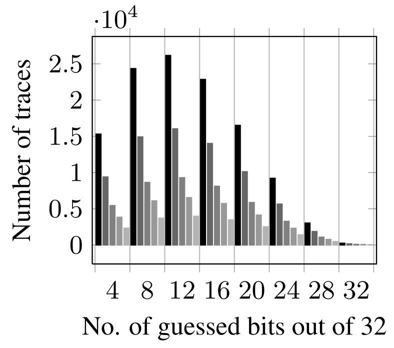
Meaningful values for α range in $0{<}\alpha{<}0.5$, as $\alpha{=}0.5$ is equivalent to a fair coin toss to decide whether the result is correct or not. Figure 4 depicts the sizes of the confidence intervals for $\alpha \in \{0.2, 0.1, 0.05, 0.01, 0.001\}$, where the lighter grey represents a greater value of α, with the number of traces n_{tr} ranging from thirty to one million.

Depending on the operation being modeled by the attacker, the distance of the theoretical value of the correlation coefficient of the correct key guess from the one most likely to be mistaken by it changes. A statistically sound criteria to actually distinguish them is that the sum of their unilateral confidence intervals is smaller than the distance between the sample correlation values. For instance, the `xor` operation, which is by far the most common way to perform KEY-ADDITIONs, has the property of having the value of the consumption hypothesis being linearly related to the ratio of correctly guessed bits over the total bits of the secret key being employed in the operation. This, in turn, implies that guessing $b{-}1$ out of b bits of the secret key during a w-bit wide KEY-ADDITION ($1{\le}b{\le}w$) will yield a distance between the sample correlation coefficients of $\delta{=}\rho_b - \rho_{b-1} = \rho_b\left(1-\sqrt{1-\frac{1}{b}}\right)$, with $\rho_b = \rho\sqrt{\frac{b}{w}}$ where ρ denotes the correlation coefficient computed guessing $b{=}w$ bits [15]. Therefore, the value of the sum of the unilateral confidence intervals for ρ_b and ρ_{b-1} should be smaller than

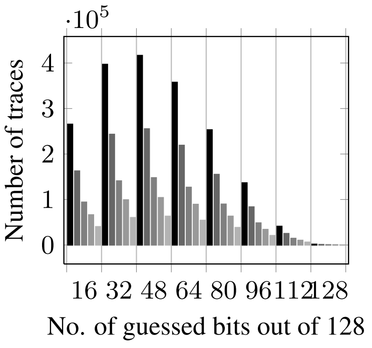
[2]$z_{1-\alpha/2}$ denotes the quantile of order $1-\alpha/2$ of the normal distribution.



(a) 8-bit datapath (w=8)

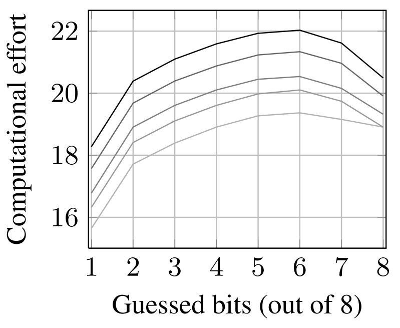


(b) 32-bit datapath (w=32)

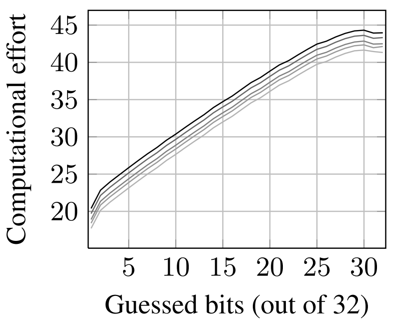


(c) 128-bit datapath (w=128)

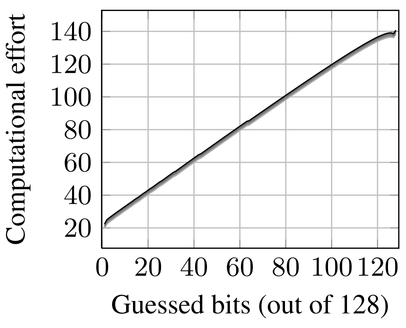
Figure 2: Expected number of traces required to lead a CPA against the 1st AddRoundKey of an hardware implementation of AES-128 as a function of the number of bits employed to formulate the key hypothesis (and of the error probability α in $\{0.2, 0.1, 0.05, 0.01, 0.001\}$) in case the device can compute the AddRoundKey with w=8, w=32 or w=128 bits at time. Lighter greys represent greater values for α



(a) 8-bit xor



(b) 32-bit xor



(c) 128-bit xor

Figure 3: Computational effort required to retrieve the full key ($k = 128$) with an attack against the xor operation, $w \in \{8, 32, 128\}$, as a function of the number of bits b involved in the key hypothesis. The computational effort is plotted in $\log_2$ scale (i.e. number of bits of equivalent security margin) and considering an error probability of $\alpha \in \{0.2, 0.1, 0.05, 0.01, 0.001\}$ (lighter gray stands for larger αs). The effective trace length to compute the complexity l is 1, as the attacker knows the exact time instant.

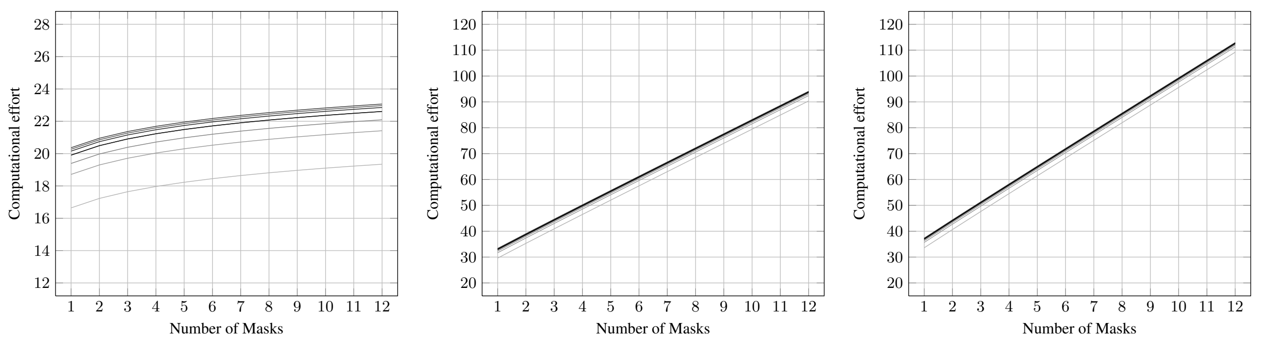
δ. Consequentially the minimum number of traces needed to led a successful CPA corresponds to the one which yields the confidence intervals small enough to match this condition.

Figure 2 depicts the values of the expected number of traces varying the number of guessed bits for 8- 32- and 128-bit wide xor operations, respectively, (i.e. the values of $\mathcal{F}(n_{tr}, b)$) considering an error probability varying in $\{0.2, 0.1, 0.05, 0.01, 0.001\}$. The bell-like shape of the plots is to be ascribed to the fact that the difference between ρ_b and ρ_{b-1} widens with the growing number of hypothesized bits, while the confidence intervals for the values of ρ_b and ρ_{b-1} shrink. In case the operation being modelled does not enjoy the aforementioned property, we note that it has a finite number of inputs (e.g. 2^8 for an 8-bit S-BOX). To determine the values of the correlation coefficients of the correct and most likely to be mistaken key hypothesis the simulated power model (which fits perfectly the circuit) can be employed, computing the correlation coefficients for each possible key hypothesis and observing which one is the most likely mistaken one. After the most likely to be mistaken key hypothesis has been determined, it is possible to simply compute the distance δ between its correlation coefficient and the one of the correct key hypothesis. Finally, this value of δ is employed as before to find out the minimum number of traces n_{tr} that is necessary to acquire to perform a successful CPA for a fixed value of the error probability α.

Definition 3.3. *Let l be the number of samples of a trace and k the total number of key bits to be retrieved, the minimum Ideal computational effort to lead a CPA is the one where the number of key bits b guessed during a single CPA minimizes the computational effort to retrieve the whole key:*

$$b = \underset{b \in \{1,\ldots,k\}}{\operatorname{argmin}} \left(\frac{k}{b} l \mathcal{F}(n_{tr}, b) \right)$$

Figure 3 depicts the amount of computational effort required to lead a successful attack against an 8-, 32- and 128-bit xor, considering as n_{tr} the minimum one to obtain the desired value of α, i.e. the ones depicted in Figure 2. As it can be noticed, raising the number of guessed bits at once raises exponentially the effort of leading the CPA: this effect dominates the linear reduction in complexity due to the fact that a smaller number of CPA attacks should be lead to recover all the k bits of the key. Note that a lower precision in knowing the position of the targeted operation makes the computational complexity scale linearly, as the number of correlation coefficients to be computed scales accordingly.



(a) Masking only (b) Masking and Hiding ($h = 16$) (c) Masking, Hiding ($h = 16$) and considering sampling

Figure 5: Computational effort required to breach a masked implementation of the 8-bit `xor` operation, considering an error probability of $\alpha = 0.2$, only. The computational effort is plotted in $\log_2$ scale (i.e. number of bits of equivalent security margin). Lighter greys indicate a lower number of guessed bits at once ($1 \leq b \leq w$)

3.1 DPA Countermeasures and High-Order Attacks

Countermeasures to prevent the leakage of power-based side-channels are split into two categories: *masking*, i.e. performing a randomized computation where the actual sensitive value is added to random masks, and *hiding*, i.e. performing the computation in a different time instant at each run of the algorithm.

3.1.1 Masking

Masking aims at invalidating the link between the predicted hypothetical power consumption values and the actual values processed by the device. In a masked implementation, each sensitive intermediate value is concealed through splitting it in a number of shares, which are processed in different time instants. Hence, the target algorithm is modified to correctly process each share and to recombine them at the end of the computation. A masking scheme with only two shares is composed by the values v_m and m, where m is a randomly chosen mask and v_m is a share such that the value v to be protected can be derived as $v = v_m \diamond m$, with $\diamond$ denoting an invertible binary operation (usually, the bitwise xor). To compensate for this countermeasure, more sophisticated DPA attacks, known as higher-order DPAs (HO-DPA) rely on predicting the consumption of the operations handling the shares and try to obtain a combination of them which is independent from the masking values. This value must subsequently be correlated with an analogous combination of the measured consumption values, employing the usual Pearson correlation coefficient. Thus, under the hypothesis of a perfect knowledge of all the time instants where the masked operations are performed, introducing an m-th order masking scheme on a cipher implies a growth of the effective length of the trace l up to $m+1$ (as enough relevant power consumption samples must be obtained). However, as the attacker performs the actual correlation analysis on the post-processed values, the complexity of the DPA increase simply by the amount of time required to perform the pre-processing action, typically a computational effort linear in the number of acquired traces. This in turn implies that a masking scheme is effectively raising by a significant margin the security of the implementation only if the attacker is not able to know exactly where the masked values are computed, i.e. he needs to sample more than $l{=}m{+}1$ points and compute the recombination function for all the possible combination of them. In the general case it holds that the attacker needs to compute $\binom{l}{m+1}$ correlation analyses for an m-th order masking scheme, thus obtaining a computational complexity of $\mathcal{O}\big(\binom{l}{m+1}\, l\, \frac{k}{b}\, 2^b\, 4\, n_{tr}\big)$. This in turn implies that the exact knowledge of the time instant when the sensitive masked operations happen is a crucial asset for an attacker as, in turn, when this knowledge is not available, the computational effort grows very quickly.

3.1.2 Hiding

Hiding methods aim to conceal the relation between the power consumption and the operations performed by the target algorithm to compute the intermediate values. This is done either through randomizing the time instant in which the sensitive operation is performed (also known as hiding in the time dimension [14]), or through adding extra dummy computation performed in parallel with the sensitive operation, thus lowering the theoretical maximum correlation (also known as hiding in power). The protection strategies employed in the open literature are based on execution flow randomization via shuffling the order of some operations (f.i., permuting the sequence of accesses to lookup tables) and inserting random delays built with dummy operations or with clock randomization methods [18, 22, 23]. To minimize the performance overhead, the execution must be interleaved with delays in multiple places, keeping the individual delays as short as possible. In this way, an attacker faces a cumulative and hardly predictable sum of delays between the start (the end, respectively) of the algorithm and the location of the observed intermediate operation in time [13]. *Hiding in time* is effective in reducing the maximum correlation obtainable by an attacker, since computing the usual correlation coefficient over n_{tr} traces, where the sensitive operation is performed every time in one out of h possible time instants will yield a reduction in the correlation coefficient by h. This reduction is due to the fact that only a single trace over h has the operation performed in the same instant, and there are $h{-}1$ unrelated samples being added to it (both these factors reduce the theoretical correlation coefficient by a factor of $\sqrt{h}$). An attack strategy to partially compensate for hiding in time [11] involves pre-processing the power trace with performing a sliding window sum of the samples, with a window length h as large as the time range in which the sensitive operation may be performed. The resulting processed signal will have a sample which will always contain the power consumption of the correct operation, added to $w - 1$ unrelated power consumption samples, thus yielding a net reduction in the theoretical correlation coefficient of $\sqrt{h}$. *Hiding in power* has the direct effect of reducing the maximum correlation between the measured power consumption

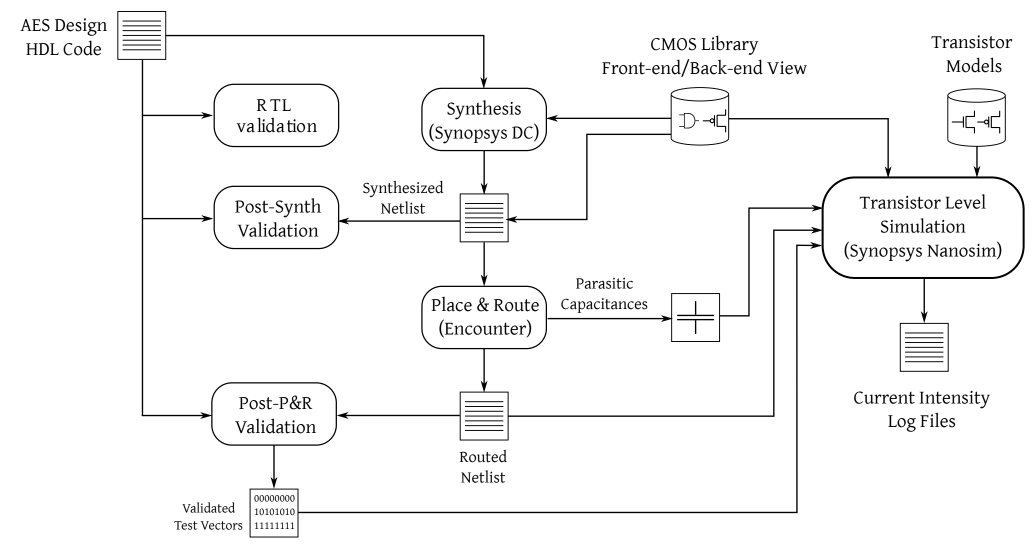


Figure 6: Detailed description of the simulation and verification tools employed to obtain the simulated power traces. The workflow is a specialization of the general one described in Figure 1

depending on the entity of the consumption of the power operation. This reduction cannot be compensated in any way: the attacker is forced to collect more traces in order to obtain a reliable estimation of a smaller correlation coefficient. The amount of this reduction can be evaluated considering the dummy operation as a component of the sensitive operation and applying the method mentioned in the previous subsection. Summing up, the net effect of applying hiding strategies to an implementation of a cipher is a $\sqrt{h}$ decrease in the correlation coefficient and an increase in the number of samples of the power consumption which should be acquired to h in order to record the power consumption of the correct operation. Consequentially the computational effort required for a DPA attack against an implementation protected with hiding techniques will rise as a consequence of the higher number of traces required to achieve a statistically sound estimation of a smaller difference in correlation coefficients.

3.1.3 Combined Strategies

The typical countermeasure solutions usually employ both hiding and masking in a combined setting to provide a higher security margin. Typically, engineering solutions limits the overhead of high-order masking schemes with respect to the unprotected implementation, resorting to the combination of two-share masking schemes and hiding techniques [18, 19] and this is commonly accepted to be reasonably secure. Willing to provide a quantitative margin of the security of employing a combined hiding and masking scheme we note that the two approaches have a strong positive synergy, as hiding masked operations raises the possible positions which the attacker should consider to compute the recombining function required for HO-DPA attacks. More in detail, given an m-th order masking scheme, the total number of power consumption samples which should be acquired is $h(m+1)$ as every masked operation is hidden within an h samples wide window. Consequentially the total computational complexity of the scheme rises up to $\mathcal{O}\left(\binom{mh+h}{m+1}\, l\, \frac{k}{b}\, 2^b\, 4\, n_{tr}\right)$. Figure 5 provides an evaluation of the computational complexity of a masked implementation, considering both the plain masking and the combined action of masking and hiding with $h = 16$.

Willing to provide an insight on an attacker which is not able to determine the perfect time instant in which the sensitive operation takes place, but is able to determine the clock cycle in which this operation happens, we recall that he should sample to at least 2.5 times the clock frequency of the device, in order to obtain the full information modulated on the clock edges [7–9, 16]. This in turn implies at least a growth factor 2.5 on the value of $h(m + 1)$, of which the effects on the security margin are depicted in Figure 5.

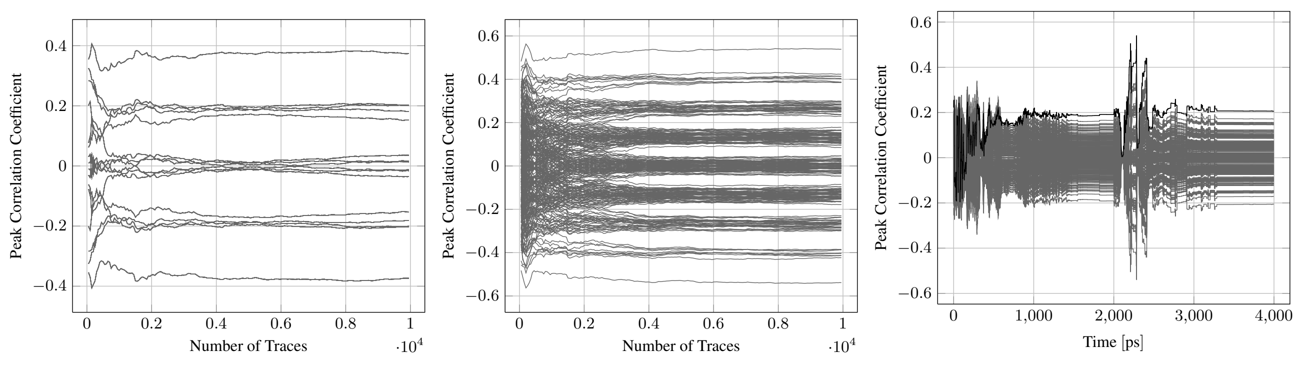
4. CASE STUDY ARCHITECTURE AND SIMULATION ENVIRONMENT

In this section we describe the architecture employed to implement the AES cipher in our case study evaluation.

The 128-bit datapath implementation of the AES is a fully parallel one where one complete round is computed at each clock cycle. To do so, there are 16 instances of the same `S-box`. The `S-box` is implemented in a combinatorial fashion, i.e. its behavior is described in HDL language and then synthesized and left to be optimized by the synthesis tool. The other primitives are performed in a straightforward parallel fashion on the whole state and only the result of the round computation is saved back in the 128-bit register at the end of the clock cycle. The initial plaintext and all the round keys were provided to the module as external signals via the HDL test bench, assuming the key schedule has already been performed during the bootstrap phase to enhance the throughput of the implementation.

Figure 6 depicts the detailed simulation setup we have employed in our evaluation procedure, following the one proposed by Regazzoni et al. in [17]. The general design flow is the one of the standard Electronic Design Automation (EDA) described in Section 2 taking as input the the Register Transfer Level (RTL) description of the considered cores obtained from the synthesis of their HDL description. The output of the simulation flow is a text file which stores the noise-free instantaneous current at the power supply line of the circuit.

The VHDL description is synthesized using the STMicroelec-



(a) 4-bit key hypothesis (b) 8-bit key hypothesis (c) Time domain representation

Figure 7: Correlation coefficients obtained estimating respectively 4 bit (Subfigure (a)) and 8-bit (Subfigure (b)) of the 128 bit XOR, together with the trend of the values of Pearson's correlation coefficient in time, estimated with 10000 traces (Subfigure (c))

tronics 65 nm GP SVT CMOS standard cell library [20] and *Synopsys Design Compiler* [21]. The place and route is carried out using Cadence Silicon Encounter and the transistor level simulation is performed using Synopsys Nanosim, simulating the environment with a time resolution of 1 ps per sample and a precision in the current measure of 1 pA. This simulation setup is the equivalent of gathering measurements on the actual circuit with an effective bandwidth of 500 GHz (taking into account Nyquist's bound) and considering a voltage probe sampling at the ends of an 1 Ω resistor for a peak-to-peak dynamic consumption variation of 4 μV with a 20-bit ADC. With the current state-of-the-art technologies, such precision in measurements is not achievable in practice, especially considering that the chip bonding wires add substantial parasitic capacitances which cannot be removed easily at measurement time. Moreover, the current measurements provided by the simulator are perfectly aligned timewise and do not suffer of any measurement noise. Taking into account these facts, we consider the quality of the measurements produced by our simulation a conservative upper bound on the measurement capability of any practical attacker targeting our implementations.

The implementation was correctly synthesized, placed and routed constraining the clock period to be 4 ns or shorter (thus supporting a working frequency of 250 MHz) and optimizing for area as much as possible within that constraint. We notice that a significant part of the 128-bit core is taken up by the `S-Box`es and `xor` gates (namely, roughly 72% of the final circuit area). Consequentially, we expect the correlation coefficients obtained from the simulation of the 128-bit core to be closer to the theoretically predicted ones.

5. METRIC EVALUATION

In order to evaluate our metric, we collected simulated measurements from both the implementation of the AES, employing as plaintexts the ones provided by the NIST test-benches, in order to be able to provide also the correct validation values to the simulator and check for the correct operation of the circuit.

Willing to evaluate the effectiveness of our metric, we collected 10000 measurements, in order to provide an estimate of the correlation coefficients obtained performing attacks against the `xor` operation employed to compute the ADDROUNDKEY primitive. According to the metric, this should be sufficient to estimate the sample correlation coefficient r for a b=8 bits of key-hypothesis on the 128-bit `xor` with error probability α as low as 0.001. The expected correlation coefficient for the correct key value is thus of $\sqrt{\frac{8}{128}} = 0.25$, assuming that all the 128 `xor` operations are performed exactly in the same instant.

The results of the actual correlation power analysis conducted on the collected measurements are depicted in Figure 7 and show that the actual sample correlation coefficient obtained on the simulated measurements is higher than the one expected and, more precisely is ≈ 0.5. Willing to explain this behavior we analyzed the post-place-and-route floorplan of the circuit and observed that the floorplanning tool clustered together banks of 32-bit `xor` gates during the synthesis, thus effectively splitting the 128-bit wide `xor` operation into four 32-bit wide ones. This layout causes the `xor` to be computed in slightly different time instants, thus resulting in an effective estimate of 8-bit out of 32 being computed, which in turn yields a theoretical correlation coefficient of $\sqrt{\frac{8}{32}} = 0.5$, which is actually coherent with our theoretical estimates. In this case, estimating a-priori the theoretical correlation coefficient allowed us to discover that the our design was not placed and routed as expected, thus yielding an effective reduction in the number of operation being performed in the same instant. We note that, taking into account the fact that only 32 `xor` operations are performed together, we still able to consider the estimates we obtain from 10000 measurements to be reliable with an error probability lower than α=0.05 in estimating the highest correlation coefficient. The results depicted in Figure 7 were computed to further confirm the fact that actually only 32 of the `xor` operation are computed at once, and, also in this case, the estimated correlation coefficients match the theoretical ones, while the error probability is reduced to α=0.01, according to Section 3.

Figure 7(c) reports the trend of Pearson's correlation coefficients with respect to time, showing where the simulated power traces report a non-negligible correlation with the key hypothesis. The figure depicts the clock cycle when the plaintext is loaded and the first AddRoundKey is performed and where the latch saves the result (the result is saved on the raising edge of the clock). As it can be noticed, the value for the correlation coefficient reaches its maximum in the time instant where the latches actually save the value of the output of the AddRoundKey (around 2000 ps), as its power consumption is linearly dependent on the Hamming weight of the output of the AddRoundKey. We also note that, thanks to the accuracy of the simulations, it is also possible to notice a non negligible correlation of the static power consumption of the circuit with the predicted consumption model. In particular, after the value being predicted is stored in the register it is possible to notice that the correlation with the static power consumption is non negligible, and the correct key emerges with a statistical confidence greater than 99%. This is consistent with what has been reported in open literature in [5] as the possibility of performing Leakage Power Analysis, i.e. exploiting the static power consumption to perform side channel attacks.

Willing to apply our security metric to the 128-bit core under exam, employing the correct model for the placed and routed implementation, following the results proposed in Section 3 we obtain that a 9th order masking scheme on the `xor` operation, with a hiding factor of 16 (i.e. each `xor` operation takes place in one out of 16 possible time instants, which is implementable with no effort through shuffling the order of the four 32-bit operations) yields an effective computational security margin greater than 80 bits, assuming that the attacker is able to choose the most advantageous computational effort. This estimate for the security margin of the cipher implementation is rather conservative, and can be brought closer to reality taking into account the $2.5\times$ penalty factor in time uncertainty brought in by the need of the attacker to sample the power consumption while keeping in mind Nyquist's bound. In this case, the effective number of masks required to achieve a 80-bit computational security margin is reduced to 7, which, albeit being higher than the usual advised number of 2 is bound to provide an effective security margin even taking into account the best possible measurement and modeling skills of the attacker against this cipher primitive.

6. CONCLUSIONS

In this work we presented a first exploration of a design-time security metric to evaluate the effective computational security margin against side-channel attacks of an implementation of a cryptographic primitive. We employed an attacker model which assumes the ability of performing perfect measurements and the complete knowledge of the implementation details by the attacker, and consequentially provided the computational complexity of a power analysis side-channel attack against the implementation. We evaluated the effect of employing hiding and masking countermeasures to raise the security margin of a design up to the point where it matches the one provided by the cipher under exam (AES-128 in our case). Finally, we evaluated practically a case study, comparing the theoretical correlation coefficient with the one obtained practically from the simulations in order to understand its vulnerability to attacks, providing a quantitative estimate of the entity of the masking and hiding needed to secure the implementation.

7. REFERENCES

[1] G. Agosta, A. Barenghi, F. De Santis, A. Di Biagio, and G. Pelosi. Fast Disk Encryption through GPGPU Acceleration. In *PDCAT*, pages 102–109. IEEE Computer Society, 2009.

[2] G. Agosta, A. Barenghi, F. De Santis, and G. Pelosi. Record Setting Software Implementation of DES Using CUDA. In *ITNG*, pages 748–755, 2010.

[3] G. Agosta, A. Barenghi, and G. Pelosi. A Code Morphing Methodology to Automate Power Analysis Countermeasures. In P. Groeneveld, D. Sciuto, and S. Hassoun, editors, *DAC*, pages 77–82. ACM, 2012.

[4] G. Agosta, A. Barenghi, and G. Pelosi. Exploiting Bit-level Parallelism in GPGPUs: a Case Study on KeeLoq Exhaustive Search Attacks. In *ARCS Workshops*, pages 385–396, 2012.

[5] M. Alioto, L. Giancane, G. Scotti, and A. Trifiletti. Leakage Power Analysis Attacks: A Novel Class of Attacks to Nanometer Cryptographic Circuits. *IEEE Trans. on Circuits and Systems*, 57-I(2):355–367, 2010.

[6] A. Barenghi, G. Bertoni, L. Breveglieri, and G. Pelosi. A FPGA Coprocessor for the Cryptographic Tate Pairing over F_p. In *ITNG*, pages 112–119. IEEE Computer Society, 2008.

[7] A. Barenghi, G. Pelosi, and Y. Teglia. Improving first order differential power attacks through digital signal processing. In O. B. Makarevich, A. Elçi, M. A. Orgun, S. A. Huss, L. K. Babenko, A. G. Chefranov, and V. Varadharajan, editors, *SIN*, pages 124–133. ACM, 2010.

[8] A. Barenghi, G. Pelosi, and Y. Teglia. Information Leakage Discovery Techniques to Enhance Secure Chip Design. In C. A. Ardagna and J. Zhou, editors, *WISTP*, volume 6633 of *Lecture Notes in Computer Science*, pages 128–143. Springer, 2011.

[9] A. Barenghi, G. Pelosi, and F. Terraneo. Secure and Efficient Design of Software Block Cipher Implementations on Microcontrollers. *Int. Journal of Grid and Utility Computing*, 2012.

[10] G. Bertoni, L. Breveglieri, L. Chen, P. Fragneto, K. A. Harrison, and G. Pelosi. A pairing SW implementation for Smart-Cards. *Journal of Systems and Software*, 81(7):1240–1247, 2008.

[11] C. Clavier, J.-S. Coron, and N. Dabbous. Differential Power Analysis in the Presence of Hardware Countermeasures. In Çetin Kaya Koç and C. Paar, editors, *CHES*, volume 1965 of *Lecture Notes in Computer Science*, pages 252–263. Springer, 2000.

[12] C. Clavier and K. Gaj, editors. *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747 of *Lecture Notes in Computer Science*. Springer, 2009.

[13] J.-S. Coron and I. Kizhvatov. Analysis and Improvement of the Random Delay Countermeasure of CHES 2009. In S. Mangard and F.-X. Standaert, editors, *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 95–109. Springer, 2010.

[14] S. Mangard. Hardware Countermeasures against DPA? A Statistical Analysis of Their Effectiveness. In T. Okamoto, editor, *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 222–235. Springer, 2004.

[15] S. Mangard, E. Oswald, and T. Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.

[16] A. Moradi, A. Barenghi, T. Kasper, and C. Paar. On the vulnerability of FPGA bitstream encryption against power analysis attacks: extracting keys from Xilinx Virtex-II FPGAs. In Y. Chen, G. Danezis, and V. Shmatikov, editors, *ACM Conference on Computer and Communications Security*, pages 111–124. ACM, 2011.

[17] F. Regazzoni, A. Cevrero, F.-X. Standaert, S. Badel, T. Kluter, P. Brisk, Y. Leblebici, and P. Ienne. A Design Flow and Evaluation Framework for DPA-Resistant Instruction Set Extensions. In Clavier and Gaj [12], pages 205–219.

[18] M. Rivain, E. Prouff, and J. Doget. Higher-Order Masking and Shuffling for Software Implementations of Block Ciphers. In Clavier and Gaj [12], pages 171–188.

[19] K. Schramm and C. Paar. Higher Order Masking of the AES. In D. Pointcheval, editor, *CT-RSA*, volume 3860 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2006.

[20] STMicroelectronics. Clock 90 GPLVT 1.2V 2.2 Standard Cell Library User Manual and Databook, Sept. 2006.

[21] Synopsis Corporation. Design Compiler Ultra. White paper Sep. 2012, `http://www.synopsys.com/tools/implementation/rtlsynthesis/dcultra/Pages/default.aspx`.

[22] S. Tillich and C. Herbst. Attacking State-of-the-Art Software Countermeasures-A Case Study for AES. In E. Oswald and P. Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 228–243. Springer, 2008.

[23] S. Yang, W. Wolf, N. Vijaykrishnan, D. N. Serpanos, and Y. Xie. Power Attack Resistant Cryptosystem Design: A Dynamic Voltage and Frequency Switching Approach. In *DATE*, pages 64–69, 2005.