RTL-PSC: Automated Power Side-Channel Leakage Assessment at Register-Transfer Level

Miao (Tony) He[1], Jungmin Park[1], Adib Nahiyan[1], Apostol Vassilev[2], Yier Jin[1], and Mark Tehranipoor[1]

Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611[1]

National Institute of Standards and Technology, Gaithersburg, MD 20899[2]

Email: {tonyhe, jungminpark, adib1991}ufl.edu, apostol.vassilev@nist.gov, {yier.jin, tehranipoor}@ece.ufl.edu

Abstract—**Power side-channel attacks (SCAs) have become a major concern to the security community due to their non-invasive feature, low-cost, and effectiveness in extracting secret information from hardware implementation of cryto algorithms. Therefore, it is imperative to evaluate if the hardware is vulnerable to SCAs during its design and validation stages. Currently, however, there is little known effort in evaluating the vulnerability of a hardware to SCAs at early design stage. In this paper, we propose, for the first time, an automated framework, named RTL-PSC, for power side-channel leakage assessment of hardware crypto designs at register-transfer level (RTL) with built-in evaluation metrics. RTL-PSC first estimates power profile of a hardware design using functional simulation at RTL. Then it utilizes the evaluation metrics, comprising of KL divergence metric and the success rate (SR) metric based on maximum likelihood estimation to perform power side-channel leakage (PSC) vulnerability assessment at RTL. We analyze Galois-Field (GF) and Look-up Table (LUT) based AES designs using RTL-PSC and validate its effectiveness and accuracy through both gate-level simulation and FPGA results. RTL-PSC is also capable of identifying blocks** * **inside the design that contribute the most to the PSC vulnerability which can be used for efficient countermeasure implementation.**[†]

Index terms— Side-channel Attacks, Leakage Assessment, Vulnerability Evaluation, Register-Transfer Level.

I. INTRODUCTION

Power side-channel attacks (SCAs) exploit the weaknesses in the hardware implementations of crypto algorithms to leak sensitive information, e.g., the encryption key, irrespective of the mathematical robustness of the algorithms. A number of SCAs namely Simple Power Analysis [1], Differential Power Analysis (DPA) [1], Correlation Power Analysis (CPA) [2], Template Attacks [3], Mutual Information Analysis (MIA) [4], and Partitioning Power Analysis (PPA) [5] have been proposed and successfully demonstrated over the past two decades. These attacks exploit the fact that the power consumption of a cryptographic hardware depends on the data it processes and the operation it performs [6].

To counter these attacks, a variety of countermeasures have been proposed to make the crypto hardware resilient to SCAs. The goal of these countermeasures is to reduce the dependencies between the intermediate values of the cryptographic algorithms and the power consumption of the cryptographic devices. For example, hiding countermeasures attempt to break the link between the processed data values and the power consumption of the devices [6]. However, all of the SCA countermeasures adversely affect the circuit area, thus making them impractical to be applied to modern resource-constrained designs. One major reason may come from the fact that the evaluation methodology for side-channel leakage assessment are not capable of identifying the source of vulnerabilities

effectively and accurately. Hence, the corresponding countermeasure has to be applied to the entire design instead of specific sub-blocks responsible for power side-channel leakage (PSC) vulnerability.

Apart from power side-channel attacks and their corresponding countermeasures, another highly important topic in this domain is power side-channel leakage assessment. Several techniques have been proposed in this domain including signal-to-noise ratio (SNR) [7], Test Vector Leakage Assessment (TVLA) methodology [8], success rate [9], and autonomous side-channel vulnerability evaluator (AMASIVE) [10], [11]. However, these techniques suffer from the following major limitations.

- They mostly focus on the post-silicon side-channel assessment, which is too late and prohibitively expensive in making any changes to the design to address the leakage issue.
- Existing techniques typically require large amount of plain-texts and power traces, hence need prohibitively large simulation time, i.e., these techniques are feasible for the post-silicon stage rather than the pre-silicon stage.
- Some techniques, e.g., TVLA [8] and χ^2-test [12] can only provide a pass/fail test and cannot give quantitative measure of PSC vulnerability which can lead to false positive results.
- Existing techniques mostly require a security analysts to be manually involved in leakage assessment, which may not be feasible due to cost and time-to-market constraints for modern devices.

We summarize the evaluation time and accuracy of side-channel leakage assessment as well as the flexibility to make design changes at different pre-silicon design stages w.r.t. the post-fabricated device level in Figure 1. In the pre-silicon stage, as the leakage assessment accuracy increases, the leakage evaluation time increases exponentially from RTL to gate-level (GTL) to layout level. It can also be observed that the flexibility in making design changes is reduced from RTL to gate-level to layout level. On the other hand, when observing the post-silicon stage, leakage assessment can be performed efficiently and accurately, however, flexibility for making design changes is very difficult (as in FPGA) if not impossible (as in ASIC).

Our Contributions: We propose a framework named RTL-PSC which can automatically assess PSC vulnerability at the earliest pre-silicon design stage, i.e., RTL. This framework is developed to be integrated into the traditional ASIC and FPGA design flow. RTL-PSC provides distinctive capabilities to chip designers and security analysts as listed below:

- **Leakage assessment at early design stage.** The RTL-PSC framework is developed to automatically evaluate PSC vulnerability of a design at higher levels of abstraction to reduce time-to-market, cost of redesign, and the overall cost of adding security to the design.
- **Technology independent.** The vulnerability analysis is performed on RTL design to make the evaluation framework fairly library independent.

---

* 'design' refers to top module as well as its constituting sub-modules, while a 'block' refers to a sub-module within the design.
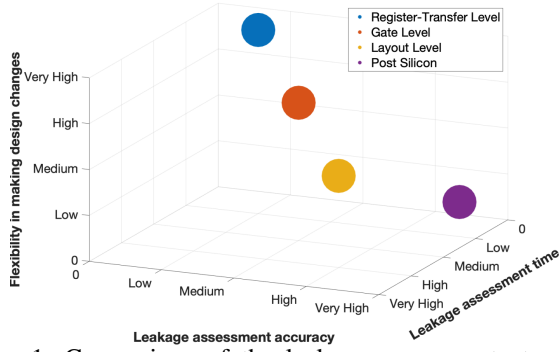
Figure 1: Comparison of the leakage assessment at various stages of the design process.

- **Fine granularity evaluation.** The RTL-PSC framework identifies which block of a design contributes the most to the vulnerability, i.e., pinpoints which block is leaking more secret information. Countermeasures only need to be applied for the vulnerable blocks/modules to reduce area overhead significantly.
- **Comprehensive analysis.** RTL-PSC considers the relation between blocks/modules of a design during vulnerability analysis instead of analyzing blocks/modules of a design independently.
- **Fast power estimation.** RTL-PSC estimates power leakage distribution based on the number of transitions at RTL to ensure the evaluation framework is fast.
- **Generic framework.** The RTL-PSC framework can be automatically applied to any cryptographic implementations at RTL without any customization or designers' involvement.
- **Time, accuracy, and flexibility trade-off.** RTL-PSC can accurately and efficiently estimate PSC vulnerability at RTL. RTL-PSC has an average evaluation time of 35*mins* and its evaluation results closely matches with silicon results obtained from FPGA. Also, RTL-PSC provides the hardware designers with the most flexibility to address PSC vulnerabilities having the capability of working at RTL.

The rest of the paper is organized as follows: In Section II, the RTL-PSC framework is presented. Section III presents the RTL side-channel vulnerability evaluation metrics. Section IV provides and analyzes in detail the results demonstrating the performance of RTL-PSC and its respective metrics proposed in this paper. Finally, concluding remarks are offered in Section V.

II. RTL-PSC VULNERABILITY EVALUATION FRAMEWORK

Figure 2 outlines the side-channel vulnerability evaluation framework. It includes two main parts, RTL Switching Activity Interchange Format (SAIF) file generation shown in the blue box and identification of vulnerable designs and blocks shown in the purple box. Algorithm 1 describes the identification technique for vulnerable designs and blocks. Note that, here TC refers to the transition count, $\mathcal{N}$ refers to the Gaussian distribution, f refers to the probability density function (PDF), D_{KL} refers to the KL divergence and ML refers to the maximum likelihood. Specifically, in Step 1, a group of simulation keys are specified. In Step 2, we utilize Synopsys VCS to perform functional simulation of the RTL design with the plaintexts and the selected keys as the inputs (key selection process is described in Section III-A). In Step 3, once the simulation is complete, the SAIF file for the RTL design is generated. After finishing Step 3, all SAIF files for a group of keys and the applied plaintexts are generated (See Algorithm 1, Lines 4-8). As its name indicates, the SAIF file includes the switching activity information for each net and register in the RTL design. Moreover, the SAIF file generated based on the RTL design has the same hierarchy as the

---

Algorithm 1 Identifying Vulnerable Blocks

```
 1: procedure IDENTIFYING VULNERABLE BLOCKS
 2:   Input: RTL design, 𝒫 = {Plaintext}, {Key₀, Key₁}
 3:   Output: Set_Vulnerable
 4:     for Keyⱼ ∈ {Key₀, Key₁} do
 5:       for Plaintextᵢ ∈ 𝒫 do
 6:         SAIFᵢ^{Keyⱼ} ← VCS(Plaintextᵢ, Keyⱼ)
 7:       end for
 8:     end for
 9:     for all Blockᵢ ∈ ℳ do
10:       TC⁰_{Blockᵢ} ← {SAIF₁^{Key₀}, …, SAIFₙ^{Key₀}}
11:       TC¹_{Blockᵢ} ← {SAIF₁^{Key₁}, …, SAIFₙ^{Key₁}}
12:       f⁰ᵢ ← 𝒩(μ_{TC⁰_{blockᵢ}}, σ²_{TC⁰_{blockᵢ}})
13:       f¹ᵢ ← 𝒩(μ_{TC¹_{blockᵢ}}, σ²_{TC¹_{blockᵢ}})
14:       KLᵢ ← D_{KL}(f⁰ᵢ, f¹ᵢ)
15:       SRᵢ ← ML(f⁰ᵢ, f¹ᵢ, n Plaintexts)
16:       if SRᵢ > SR_{threshold} or KLᵢ > KL_{norm.th} then
17:         Set_Vulnerable ← Blockᵢ
18:       end if
19:     end for
20: end procedure
```

design itself, hence, in Step 4, the SAIF file for each module in the design can be separated for localized vulnerability analysis. Next, the evaluation metrics are applied for leakage assessment. Specifically, in Step 5, the obtained switching activity is exploited to estimate the power leakage distribution for the design and each module within it (Lines 10-11 in Algorithm 1). In Step 6, the Kullback-Leibler (KL) divergence [13] and success rate (SR) based on power leakage distribution are calculated for the design and each block (Lines 12-15 in Algorithm 1). In Step 7, vulnerability analysis is performed for the design and each block. In Step 8, the vulnerable design is identified based on the analysis performed in the previous step. Then the vulnerable blocks in the design that are leaking information the most are identified for further processing. Step 7 and Step 8 correspond to Lines 16-18 in Algorithm 1. Following this, the framework enters into Step 9, where countermeasures only need to be applied to the vulnerable block(s). Note that Step 9 is outside the scope in this paper.

III. EVALUATION METRICS

In order to reduce time-to-market and the overall cost of adding security to the design, the RTL power side-channel vulnerability evaluation metrics are proposed to perform a design-time evaluation of PSC vulnerability. *To be specific, Kullback-Leibler (KL) divergence metric and success rate (SR) metric based on maximum likelihood estimation are developed and combined to evaluate vulnerability of a hardware implementation.* The first evaluation metric, i.e., KL divergence metric, estimates statistical distance between two different probability distributions, which is defined as follows [13]:

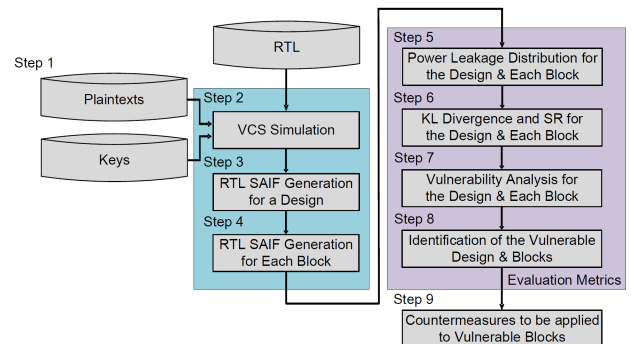$$D_{KL}(k_i||k_j) = \int f_{T|k_i}(t) \log \frac{f_{T|k_i}(t)}{f_{T|k_j}(t)} dt \quad (1)$$

Figure 2: RTL-PSC framework.

where $f_{T|k_i}(t)$ and $f_{T|k_j}(t)$ are the probability density functions of the switching activity given keys k_i and k_j, respectively.

For instance, if power leakage probability distributions based on two different keys are distinguishable, KL divergence between these two distributions is high, which provides indication on how vulnerable the implementation is. Hence, KL divergence is suitable for the vulnerability comparison between different implementations. However, the KL divergence value may be difficult to interpret when performing vulnerability analysis for just one implementation. To address this issue, we introduce the second evaluation metric, named success rate (SR)[‡] metric based on maximum likelihood estimation. The SR value represents the probability to reveal the correct key and is suitable to evaluate vulnerability for only one implementation. We derive the SR metric as follows: we assume that the probability density function of the switching activity T given a key K follows a Gaussian distribution, which can be expressed as follows:

$$f_{T|K}(t) = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{(t-\mu_k)^2}{2\sigma_k^2}} \tag{2}$$

where μ_k and σ_k^2 are the mean and variance of T, respectively. The likelihood function is defined as $\mathcal{L}(k;t) = \frac{1}{n}\sum_{i=1}^{n}\ln f_{T|K}(t_i)$. Based on the maximum likelihood estimation, an adversary typically selects a guess key $\hat{k}$ as follows:

$$\hat{k} = \underset{k \in K}{\arg\max}\, \mathcal{L}(k;t) = \underset{k \in K}{\arg\max}\, \frac{1}{n}\sum_{i=1}^{n}\ln f_{T|K}(t_i) \tag{3}$$

If the guess key ($k_g = \hat{k}$) is equal to the correct key (k^*), the side-channel attack is successful. Thus, the success rate can be defined as follows:

$$SR = \Pr[k_g = k^*] = \Pr[\mathcal{L}(k^*;t) > \mathcal{L}(\langle\bar{k^*}\rangle;t)] \tag{4}$$

where $\langle\bar{k^*}\rangle$ denotes all wrong keys, i.e., the correct key k^* is excluded from $\{k_1, k_2, \ldots, k_{n_k-1}\}$.

KL divergence is closely related to SR since the mathematical expectation of $\mathcal{L}(k^*;t) - \mathcal{L}(k_i;t)$ in Equation (4) is equal to KL divergence between $T|k^*$ and $T|k_i$ [14]. Hence, SR increases accordingly as KL divergence increases. Due to the relation between KL divergence and SR based on maximum likelihood estimation, the combination of KL and SR is proposed for leakage assessment.

A. Selection of a Key Pair

As shown in Algorithm 1, first, a key pair is specified, then the probability distributions of the switching activity based on that key pair can be estimated using Equation (2). The best key pair among all possible pairs is expected to provide the maximum KL divergence for vulnerability evaluation in the worst-case scenario. However, it is impossible and impractical to find the best key pair since the key space is huge, i.e., $\binom{2^{128}}{2}$. Alternatively, an appropriate key pair is able to be chosen, which satisfies the following conditions:

1) Assuming that each set of plaintexts is randomly generated, each key consists of the same subkey, e.g., $Key_0 = \{subkey \ldots subkey\} = 0x151515 \ldots 15$.
2) Hamming distance (HD) between two different subkeys is maximum, i.e., $subkey$ and $\overline{subkey}$.
3) If $D_{KL}(Key_0 || Key_i)$ increases asymptotically as i increases, $i = 1, \ldots n$, Key_0 and Key_n are the appropriate key pair, where Key_i is defined as $[subkey \ldots \underbrace{\overline{subkey}\ \overline{subkey}}_{i\ times}]$.

These key pairs can be used for the post-silicon validation. While it is difficult to measure the isolated leakage of each block by a random key pair at the post-silicon stage, the leakage of each block can be measured at the pre-silicon stage through applying the key pairs satisfying the above conditions. Moreover, the evaluation metrics would create the worst-case scenario through applying the key pairs with the maximum Hamming distance.

The selected keys applied to an AES RTL design are 16 pairs of keys starting from all $0s$ key until all Fs key. Each key has 128-bit and Hamming-distance between Key_i and Key_{i+1} is eight, which is shown in Table I. Also, to take into account not only the key's impact on the power consumption, but also the plaintext's impact on power consumption, we use one thousand random plaintexts with the selected key pairs. We use the AES cipher operation itself for the generation of pseudo random plaintext [15]. We use $Plaintext_0$ as seed and then use each ciphertext (j) as the next plaintext ($j+1$),[§]

$$Plaintext_0 = 000\ldots000$$
$$Plaintext_{j+1} = AES(Key_i, Plaintext_j), j = 0, 1, \ldots, 999. \tag{5}$$

It can be noted that the key pair Key_0 and Key_{16} in Table I satisfies the above conditions. Furthermore, it can be seen that Key_0 would create a state similar as the reset state of the design, hence, Key_0 and Key_{16} would create the worst case scenario. Therefore, this key pair is used for both the evaluation and the validation metrics, which is shown in Section IV.

Table I: Keys used in RTL-PSC framework.

Key_0	0x0000_0000_0000_0000_0000_0000_0000_0000
Key_1	0x0000_0000_0000_0000_0000_0000_0000_00FF
......	
Key_{15}	0x00FF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF
Key_{16}	0xFFFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF

B. Identification of Vulnerable Designs and Blocks

When the appropriate key pair is applied to a design, the same subkey patterns will be propagated to the same blocks, e.g., Sbox blocks in the AES design. The switching activity of each block and the entire design is recorded into SAIF files using VCS functional simulation, which corresponds to power leakage at RTL. The higher the difference between two power leakage distributions is, the higher impact the key has on the power consumption of the design/blocks, the more susceptible the design/blocks are to power analysis attack. Using the KL divergence and SR metrics, the vulnerable design and blocks within the design can be identified. If KL divergence or SR of any design or block is greater than $KL_{threshold}$ or $SR_{threshold}$, the design or the block is considered to be the vulnerable one (Line 16 in Algorithm 1). $SR_{threshold}$ is determined based on the security constraint, e.g., 95% SR with n plaintexts, while $KL_{threshold}$ is determined based on the $SR_{threshold}$ through the relation between KL divergence and SR.

IV. RESULTS AND ANALYSIS

In this section, we perform side-channel vulnerability assessment of two different implementations of AES algorithm using RTL-PSC. First, we provide a brief description of the two AES designs: AES Galois Field (GF) and AES Lookup Table (LUT). Then we present the evaluation results generated by RTL-PSC. We validate the accuracy of RTL-PSC evaluation results using gate-level simulation and FPGA silicon results.

A. AES Benchmarks

RTL-PSC framework is applied to AES-GF [16] and AES-LUT [17] encryption designs. Both are open-source designs. In AES-GF design, the AES key expansion and AES round operations occur in parallel. The AES-GF implementation takes 10 clock cycles to encrypt each data block. In contrast, the AES-LUT design first performs the key expansion and stores the expanded key in the key registers. The key expansion takes place once for each key. After the key expansion, the round operation starts and takes 11 clock cycles to encrypt a plaintext, precisely, one clock cycle for XORing a plaintext and the key, and 10 clock cycles for 10 round operations. Furthermore, the AES-GF architecture implements the AES SubByte operation with Galois-field arithmetic, while the AES-LUT architecture implements the AES SubByte operation with a lookup table.

The hierarchy of AES-GF and AES-LUT designs is as follows. The AES-GF consists of five SubByte blocks and four MixColumn blocks. Each SubByte block includes four Sbox blocks, each of which includes a GFinvComp block. On the other hand, the AES-LUT cipher consists of a SubWord block, a SubByte block and four MixColumn blocks. SubWord block performs the key expansion operation and therefore, is not related with the plaintext and it is not considered as a potential vulnerable block in the following analysis.

B. Analyzing Suitability of the Key Pairs

We first analyze that the key pairs selected for RTL-PSC evaluation adheres the conditions described in Section III-A. Figures 3(a) and 3(b) illustrate the leakage assessment results at RTL. The switching activities of all blocks during 11 clock cycles are calculated. In the AES-GF implementation, the KL divergence of the design (i.e., AES_GF_ENC) and each block increases asymptotically with increasing the Hamming distance of the key pairs as shown in Table I. Similarly, in the AES-LUT implementation, the KL divergence of the design (i.e., AES_LUT_ENC) and each block also increases asymptotically as the Hamming distance of the key pairs increases. It can be observed that the specified key pair ($key_0 = 0x00...00, key_{16} = 0xFF...FF$) satisfies the key selection conditions, hence is appropriate for RTL side-channel leakage assessment.

C. RTL Evaluation Metrics

In order to determine if a RTL design is vulnerable, KL divergence of the design per clock cycle is calculated. Figure 4(a) shows KL divergence from the second clock cycle to the 11th clock cycle of both AES-GF and AES-LUT RTL implementations, during which 10 round operations for an encryption are performed. At the second clock cycle corresponding to the first round operation, which is mostly exploited by power analysis attack, KL divergence of AES-GF and AES-LUT implementations are 0.47 and 0.28, respectively. These values correspond to 95% SR_{em}¶ vulnerability level ($SR_{threshold}$) with 25 *plaintexts* and 35 *plaintexts*, respectively, as shown in Figure 4(b). Based on KL divergence metric, as shown in Figure 4(a), KL divergence of AES-GF implementation at the second clock cycle (i.e., 0.47) is greater than that of AES-LUT implementation (i.e., 0.28). Likewise, based on SR metric, as shown in Figure 4(b), the DPA attack success rate of AES-GF implementation is higher than that of AES-LUT implementation with the same number of plaintexts. Similarly, the number of plaintexts required for successful DPA attack on AES-GF implementation is less than that on AES-LUT implementation. Hence, compared with AES-LUT

¶The SR_{em} represents the empirical SR based on actual SCA attacks with n plaintexts.

implementation, AES-GF implementation is identified as the more vulnerable design.

Vulnerable Block Identification: Once a RTL deign is determined as a vulnerable one, the next step is to identify the vulnerable blocks within the design that contribute to side-channel leakage significantly. Side-channel vulnerability can be evaluated in both time and spatial/modular domains. In other words, the evaluation metrics based on the switching activity of each block within the design are calculated at fine-granularity scale so that vulnerable blocks can be identified per clock cycle.

First, KL divergence in both time and spatial/modular domains is normalized, i.e., KL divergence of each block is divided by the maximum KL divergence of those blocks ($KL_{norm.th} = KL_i/max(KL_i)$). Then, if the normalized KL divergence of any block is greater than $KL_{norm.th} = 0.5$, that block is identified as the vulnerable one and included into the set of vulnerable blocks. Figure 5 shows KL divergence of each block in both time and spatial/modular domains. The identified vulnerable blocks (i.e., KL divergence greater than $KL_{norm.th} = 0.5$) are denoted with blue bars. Specifically, in the AES-GF design, GFinvComp blocks within Sbox0 and Sbox1 blocks are identified as the vulnerable ones; in the AES-LUT design, SubByte blocks are identified as the vulnerable ones. It should be noted that the threshold values ($KL_{threshold}$ and $KL_{norm.th}$) can be adjusted by the SR vulnerability level.

Evaluation Time: The evaluation time of RTL-PSC includes VCS functional simulation time of the RTL design as well as the data processing time required for analyzing the SAIF files. The evaluation time of RTL-PSC for AES-GF is 46.3 minutes and for AES-LUT is 24.03 minutes. If the same experiments were performed at gate-level designs, it would take around 31 hours. Therefore, our RTL-PSC is almost 42X more efficient as compared to similar gate-level assessment. The evaluation time at layout level is going to be even more expensive (more than a month). RTL-PSC also provides the flexibility to make design changes whereas, the post-silicon assessment provides no flexibility. In the next subsection, we will validate that RTL-PSC is not only efficient, but also accurate using the post-silicon results.

D. Validation of RTL-PSC

The RTL-PSC results are validated through both gate-level and FPGA implementations. We present that the PSC vulnerability assessment results generated by RTL-PSC is highly correlated to the assessment results retrieved from gate-level and FPGA.

GTL Validation: For gate-level validation, we first synthesize the RTL codes of AES-GF and AES-LUT to gate-level netlist using Synopsys Design Compiler [18] with Synopsys standard cell library. Next we utilize VCS to perform functional simulation of the netlist with the same plaintexts and keys as used in RTL, and generate the corresponding SAIF files. Then, we use Synopsys PrimeTime [18] as well as the generated SAIF files to report the power consumption for the entire design and each block inside the design. Finally, we derive the gate-level KL divergence metric for the design and each block using Equation 1.

We then calculate the Pearson correlation coefficient between the KL divergence of the RTL and GTL design/blocks (shown in Column 2 of Table III and Table II). The high correlation coefficient values ($> 90\%$) indicate that the PSC vulnerability assessment results generated by RTL-PSC at RTL is almost as accurate as the assessment results retrieved from GTL.

Next we validate the vulnerable block identification results produced by RTL-PSC. Table II presents the correlation coefficient values between the KL divergence metric for each block at RTL and gate-level. The KL metric for each block of
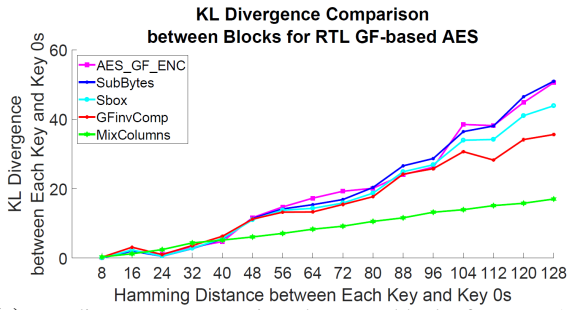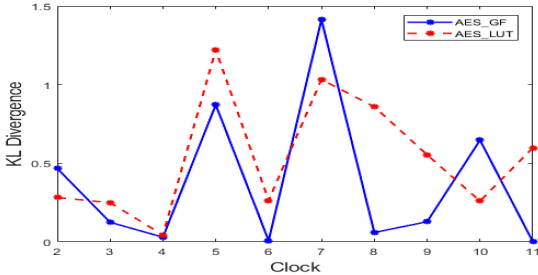
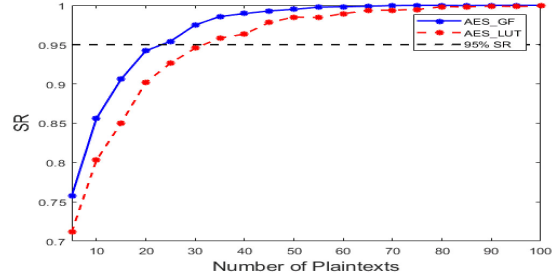(a) KL divergence comparison between blocks for RTL AES-GF implementation
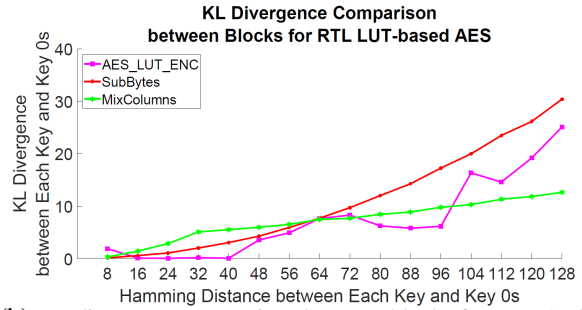
(b) KL divergence comparison between blocks for RTL AES-LUT implementation

Figure 3: KL divergence comparison between blocks for RTL AES-GF and AES-LUT implementations.
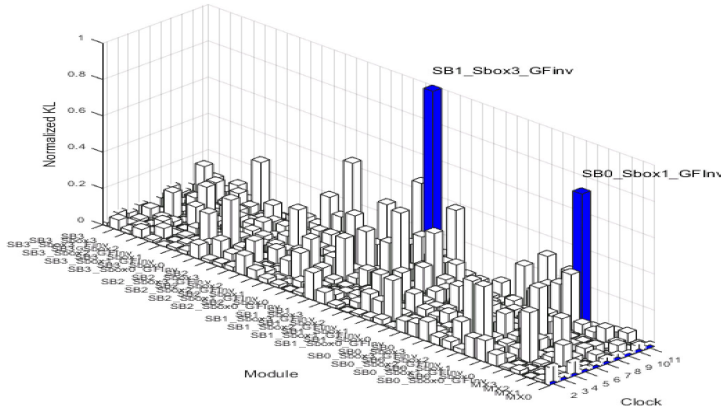

(a) KL divergence per clock cycle for AES-GF and AES-LUT implementations
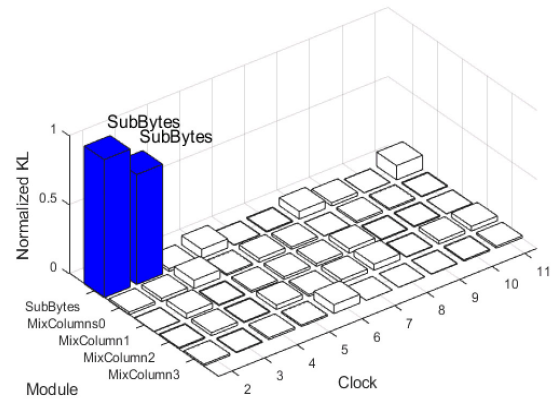

(b) SR_{em} corresponding to KL divergence (0.47 and 0.28) for AES-GF and AES-LUT implementations

Figure 4: KL divergence and SRs for AES-GF and AES-LUT implementations.

(a) Normalized KL divergence for AES-GF implementation in both time and spatial/modular domains

(b) Normalized KL divergence for AES-LUT implementation in both time and spatial/modular domains

Figure 5: Normalized KL divergence for vulnerable blocks within AES-GF and AES-LUT implementations ($KL_{norm.th} = 0.5$).

AES-GF and AES-LUT implementations at RTL has a high correlation to that at gate-level indicating that our vulnerable block identification technique at RTL is as accurate as gate-level. Next we validate the RTL-PSC evaluation results w.r.t. FPGA.

FPGA Validation: For FPGA silicon validation, we use the SAKURA-G board [19] for AES implementations, which contains two SPARTAN-6 FPGAs and is designed for research and development on hardware security. Tektronix MDO3102 oscilloscope is used to measure the voltage drop between shunt registers connected to the Vdd pin. The clock frequency of the AES implementation is 24 MHz. The sampling rate and bandwidth of the oscilloscope are 500 MS/s and 250 MHz, respectively. Figure 6 shows the experimental setup for the FPGA validation.

We first map the AES-GF and AES-LUT designs on an FPGA and then, apply the same plaintexts and keys as used at RTL, and measure the power consumption during encryption operation. Following this, we derive the KL divergence metric from the collected power traces. Then, we can calculate the Pearson correlation coefficient between the KL divergence at RTL and FPGA (as shown in Column 3 of Table III). The high correlation coefficient values ($> 80\%$) indicate that the PSC vulnerability assessment results generated by RTL-PSC at RTL is almost as accurate as FPGA assessment. In other words, RTL-PSC can accurately analyze PSC vulnerability at RTL.

Note that many implementation details, e.g., glitches caused by the gate delay, clock gating, datapath gating, retiming, clock and power network structure is not available at RTL unlike gate-level and FPGA implementations. In spite of these limitations, RTL-PSC is able to accurately identify PSC

Table II: Correlation coefficient between KL divergence of RTL and GTL blocks.

AES-GF Blocks, RTL vs. GTL				AES-LUT Blocks, RTL vs. GTL	
SubByte	Sbox	GFinvComp	MixColumn	SubByte	MixColumn
99.11%	99.55%	99.64%	94.73%	99.71%	96.80%

Table III: Correlation coefficient between KL divergence at RTL, GTL, and FPGA silicon level.

Benchmark	RTL vs. GTL	RTL vs. FPGA Silicon Level
AES-GF	99.57%	98.83%
AES-LUT	90.35%	80.80%

vulnerability which proves the efficacy of the framework.

Also, note that it is not feasible to perform vulnerable block identification at FPGA. The reason is that, it is not feasible to isolate the power traces associated to each block from the post-silicon power measurements. Therefore, we could not validate vulnerable block identification at FPGA.

Comparison with State-of-the-art: Veshchikov *et al.* [20] presented a comprehensive survey of simulators for side-channel analysis, e.g., PINPAS [21], SCARD [22], OSCAR [23], etc. These simulators mostly support software crypto algorithms implemented in microprocessors, not hardware crypto modules. On the other hand, TVLA [8] and χ^2-test [12] can work with hardware crypto designs. However, these techniques only provide a pass/fail test and do not provide the quantitative amount of leakage. AMASIVE framework [10] identifies the hypothesis function for HW/HD model to be used for side-channel vulnerability assessment. The major limitation is that it can only identify the hypothesis function and the final vulnerability assessment still needs to be carried out on a prototype device. In contrast, RTL-PSC can quantitatively and accurately assess PSC leakage of hardware crypto modules in RTL level in time efficient manner.

V. Conclusion and Future Work

In this paper, we proposed an automatic evaluation framework to perform a design-time evaluation of side-channel attacks resistance for a cryptographic implementation at RTL. Instead of measuring dynamic power, switching activity at RTL is exploited by using VCS functional simulation to estimate power profile to ensure the evaluation framework is efficient, effective, and library independent. Once the power estimation is complete, the KL divergence metric and SR metric based on maximum likelihood estimation are combined to identify the vulnerable design and blocks within the design. Experimental results on AES-GF and AES-LUT implementations demonstrated that the methodology proposed in this paper were able to perform leakage assessment and identify the vulnerable design/blocks efficiently, effectively, and precisely.

In our future work, the proposed evaluation framework will be applied to evaluate vulnerabilities of the DPA protected designs, thus providing information in terms of scaling and effectiveness for leakage assessment of protected ones. RTL-PSC will be an important component of the recent academic and industrial initiatives for developing CAD frameworks which aim at automating the security vulnerability assessment of hardware designs at design stages [24–28].
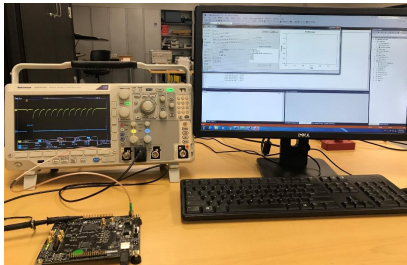

Figure 6: Experimental setup for FPGA validation.

References

[1] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '99, 1999, pp. 388–397.

[2] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems - CHES 2004*, M. Joye and J.-J. Quisquater, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 16–29.

[3] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Cryptographic Hardware and Embedded Systems - CHES 2002*, B. S. Kaliski, ç. K. Koç, and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 13–28.

[4] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel, "Mutual information analysis," in *Cryptographic Hardware and Embedded Systems – CHES 2008*, E. Oswald and P. Rohatgi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 426–442.

[5] T.-H. Le, J. Clédière, C. Canovas, B. Robisson, C. Servière, and J.-L. Lacoume, "A proposition for correlation power analysis enhancement," in *CHES 2006*, L. Goubin and M. Matsui, Eds., 2006.

[6] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[7] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining smart-card security under the threat of power analysis attacks," *IEEE Trans. Comput.*, vol. 51, no. 5, pp. 541–552, May 2002.

[8] G. Becker, J. Cooper, E. DeMulder, G. Goodwill, J. Jaffe, G. Kenworthy, T. Kouzminov, A. Leiserson, M. Marson, P. Rohatgi *et al.*, "Test vector leakage assessment (TVLA) methodology in practice," in *International Cryptographic Module Conference*, vol. 1001, 2013, p. 13.

[9] B. Gierlichs, K. Lemke-Rust, and C. Paar, "Templates vs. stochastic methods," in *Cryptographic Hardware and Embedded Systems - CHES 2006*, L. Goubin and M. Matsui, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 15–29.

[10] S. A. Huss, M. Stöttinger, and M. Zohner, "Amasive: an adaptable and modular autonomous side-channel vulnerability evaluation framework," in *Number Theory and Cryptography*. Springer, 2013, pp. 151–165.

[11] S. A. Huss and O. Stein, "A novel design flow for a security-driven synthesis of side-channel hardened cryptographic modules," vol. 7, no. 1. Multidisciplinary Digital Publishing Institute, 2017, p. 4.

[12] A. Moradi, B. Richter, T. Schneider, and F.-X. Standaert, "Leakage detection with the x2-test," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, no. 1, pp. 209–237, 2018.

[13] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, no. 1, pp. 79–86, 03.

[14] Y. Fei, A. A. Ding, J. Lao, and L. Zhang, "A statistics-based fundamental model for side-channel attack analysis," Cryptology ePrint Archive, Report 2014/152, 2014.

[15] S. S. Keller, "Nist-recommended random number generator based on ansi x9. 31 appendix a. 2.4 using the 3-key triple des and aes algorithms," *NIST Information Technology Laboratory-Computer Security Division, National Institute of Standards and Technology*, 2005.

[16] A. Lab., "Galois field based aes verilog design," 2007. [Online]. Available: http://www.aoki.ecei.tohoku.ac.jp/crypto

[17] S. Lab., "Lookup table based aes verilog design," 2017. [Online]. Available: http://satoh.cs.uec.ac.jp/SAKURA/hardware/SAKURA-G.html

[18] "Synopsys," http://www.synopsys.com/, 2018, accessed: 2018-04-20.

[19] S. Lab., "Sakura-g fpga board," 2013. [Online]. Available: http://satoh.cs.uec.ac.jp/SAKURA/hardware/SAKURA-G.html

[20] N. Veshchikov and S. Guilley, "Use of simulators for side-channel analysis," in *Security and Privacy Workshops (EuroS&PW), 2017 IEEE European Symposium on*. IEEE, 2017, pp. 104–112.

[21] J. den Hartog, J. Verschuren, E. de Vink, J. de Vos, and W. Wiersma, "Pinpas: a tool for power analysis of smartcards," in *IFIP International Information Security Conference*. Springer, 2003, pp. 453–457.

[22] M. Aigner, S. Mangard, F. Menichelli, R. Menicocci, M. Olivieri, T. Popp, G. Scotti, and A. Trifiletti, "Side channel analysis resistant design flow," in *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*. IEEE, 2006, pp. 4–pp.

[23] C. Thuillet, P. Andouard, and O. Ly, "A smart card power analysis simulator," in *Computational Science and Engineering, 2009. CSE'09. International Conference on*, vol. 2. IEEE, 2009, pp. 847–852.

[24] K. Xiao, A. Nahiyan, and M. Tehranipoor, "Security rule checking in ic design," *Computer*, vol. 49, no. 8, pp. 54–61, 2016.

[25] A. Nahiyan, K. Xiao, D. Forte, and M. Tehranipoor, "Security rule check," in *Hardware IP Security and Trust*. Springer, 2017, pp. 17–36.

[26] A. Nahiyan, K. Xiao, K. Yang, Y. Jin, D. Forte, and M. Tehranipoor, "Avfsm: a framework for identifying and mitigating vulnerabilities in fsms," in *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 2016, p. 89.

[27] A. Nahiyan, M. Sadi, R. Vittal, G. Contreras, D. Forte, and M. Tehranipoor, "Hardware trojan detection through information flow security verification," in *Test Conference (ITC), 2017 IEEE International*. IEEE, 2017, pp. 1–10.

[28] A. Nahiyan, F. Farahmandi, P. Mishra, D. Forte, and M. Tehranipoor, "Security-aware fsm design flow for identifying and mitigating vulnerabilities to fault attacks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.