Leverage the Average: Averaged Sampling in Pre-Silicon Side-Channel Leakage Assessment

Pantea Kiaei, Zhenyuan Liu, and Patrick Schaumont. 2022. Leverage the Average: Averaged Sampling in Pre-Silicon Side-Channel Leakage Assessment. In Proceedings of the Great Lakes Symposium on VLSI 2022 (GLSVLSI '22), June 6–8, 2022, Irvine, CA, USA. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3526241.3530337

ABSTRACT
Pre-silicon side-channel leakage assessment is a useful tool to identify hardware vulnerabilities at design time, but it requires many high-resolution power traces and increases the power simulation cost of the design. By down-sampling and averaging these high-resolution traces, we show that the power simulation cost can be considerably reduced without significant loss of side-channel leakage assessment quality. We introduce a theoretical basis for our claims. Our results demonstrate up to 6.5-fold power-simulation speed improvement on a gate-level side-channel leakage assessment of a RISC-V SoC. Furthermore, we clarify the conditions under which the averaged sampling technique can be successfully used.

CCS CONCEPTS
• Security and privacy → Hardware security implementation;
• Hardware → Best practices for EDA.

KEYWORDS
Pre-silicon Leakage Assessment, Power Side-Channel, Power Simulation, ASIC

1 MOTIVATION
Power side-channel attacks (PSCA) are a threat to computing systems and it is imperative that vulnerabilities of hardware products to PSCA are detected as early as possible during a chip design. To study such vulnerabilities, a designer must obtain high-resolution power traces of the design for a large number of test-vectors, and apply statistical analysis on those power traces. Two common solutions to early side-channel leakage assessment are hardware prototyping and simulation. Each solution presents unique challenges as they make a different trade-off between the speed of collecting high-resolution power traces and the fidelity of their predicted power traces to those of the actual chip. Hardware prototyping using FPGAs is a popular technique that can be applied as soon as the RTL code is available. The power traces of the actual chip under development can be predicted by measuring the power traces of the FPGA. However, ensuring fidelity to the actual chip power traces is a challenge because of the fundamental difference between the building blocks of FPGAs (configurable logic) and those of ASIC designs (standard cell libraries). Power simulation of the actual gate-level netlist using CAD tools can ensure a higher fidelity to the actual chip under development. Furthermore, the simulated power traces are noiseless and thus represent the observation of the best-equipped attacker. A major disadvantage of simulation is that the power simulation time increases drastically at the lower abstraction levels of design. Furthermore, simulation-based power side-channel leakage assessment can show the presence of leakage in a design, however, it cannot guarantee the absence of leakage.

In this paper, we describe a technique to decrease the power simulation time without raising the abstraction level of the simulation. Our method is to reduce the sample rate of the simulated power traces of a design, from one sample per clock cycle down to one sample over multiple clock cycles. Each of the resulting power samples thus represents the average power consumption over multiple clock cycles. We observe that reducing the sample rate in this manner significantly decreases the power simulation time. In addition, we show that the resulting traces at a reduced sample rate still return meaningful side-channel leakage assessment results. We derive the precise conditions under which this averaged sampling technique can be applied to side-channel leakage assessment. We also apply our results to several realistic case studies including a hardware crypto-coprocessor and a pipelined RISC-V processor.

Power simulation is commonly performed at three different abstraction levels: RTL, gate-level (post-synthesis/post-layout), and transistor-level. RTL and gate-level power simulations use event-driven simulation traces in combination with power estimators, while transistor-level power simulations are based on analog simulation. We apply our averaged sampling methodology at the gate-level, as it provides a middle ground with manageable simulation time and sufficient accuracy [17]. For example, using Composite Current Source (CCS) delay model in synthesis has shown to generate power and timing estimates comparable to that of the analog transistor-level [4]. In addition, the fidelity of gate-level power simulation can be increased by replacing the post-synthesis Standard Delay Format (SDF) file for a post-layout SDF file, generated after place and route of the chip under development, without any changes to the power simulation.

We study the averaged sampling technique as follows. In the next section, we review related work in pre-silicon side-channel leakage assessment. In section 3, we describe the basic concepts...
of averaged sampling and explain its effect on Correlation Power Analysis (CPA), a technique for side-channel leakage assessment. Section 4 introduces our case studies, including a pipelined RISC-V, and a hardware coprocessor design. We then conclude the paper.

2 RELATED WORK

At the early stages of a hardware design, after behavioral simulation/verification, power side-channel leakage (PSCL) can be investigated at the RTL-level to find and harden vulnerable parts of the design. For this purpose, both RTL-PSC [7] and Param [8] use the switching activity from RTL simulation to model the power consumption of a design.

Sijačić et al. [12] introduce a simplified model of power consumption called marching stick model (MSM) for faster less accurate modeling of power side-channel. As they point out, this model is useful for RTL-level simulations and can reduce the trace generation time significantly. However, it cannot capture the effects modeled by CAD tools in post-synthesis and later stages of the design. Furthermore, they demonstrate the most time-consuming part of the PSCL at design time is the power simulation using CAD tools.

The use of CAD tools for gate-level simulation is adopted in ACA [16] and Karna [13]. ACA finds the leaky gates in the gate-level netlist and replaces them locally with hardened structures. Karna finds the leaky parts of the implementation and replaces each gate with their lower-power versions if available in the standard cell library. Recently, the authors of Saidoyoki [10] showed simulated power traces can provide sufficient information for successful attacks with very few downsampled traces which greatly reduces the power simulation time as the bottleneck of using power simulation CAD tools in PSCL assessment at design-time.

Most pre-silicon PSCL assessment tools have explored the different abstraction layers (RTL, gate-level, transistor-level) [3]. To the best of our knowledge, our work is the first paper to study power simulation cost from the dimension of time resolution and its implication on pre-silicon PSCL assessment. As increasingly more researchers are emphasizing the integration of PSCL evaluation into ASIC design flow, it is crucial to study the implication of simulated power traces using CAD tools on leakage analysis.

3 THEORETICAL BACKGROUND

3.1 Power Side-Channel Analysis

Problem Statement. PSCAs find sensitive information processed in a device by a divide and conquer approach. Even cryptographic modules with long secret keys are susceptible to such attacks. In AES-128 (128-bit secret key) for instance, a PSCA can focus on finding one byte of the key at a time and therefore reduce the search space from $2^{128}$ to $16 \times 2^8$. In our evaluations, we focus on CPA as the most prominent form of PSCA. However, the same conclusions can be made for other statistical moment-based measures of leakage such as DPA [11] and TVLA [1]. We specifically avoid TVLA in our evaluations to prevent the known false-positive issues related to this test [14] to affect our conclusions.

CPA. Correlation power analysis (CPA) [2] finds all the subkeys processed on a device by assuming a leakage model for the device. Through calculating the Pearson correlation coefficient ($\rho$) between the collected power dissipation from the device for random known
a Value Change Dump (VCD) format. Next, the VCD stimuli are analyzed for the gate-level netlist to obtain per-frame power traces. We observe that a larger frame window size may shorten the power estimation time, and illustrate this effect in Table 1. Three different designs containing one, eight or thirty-two 32-bit counters are implemented in SkyWater 130nm standard cells. Next, their power consumption over a 10-million clock cycle testbench is estimated by Cadence Joules (v20.01). We computed traces containing 1, 100, and 1000 frames. Table 1 shows a strong dependence of power estimation time over design size, which is expected: larger designs contain more events, and therefore take more time for power estimation. However, Table 1 also shows a strong dependence over the number of frames. Indeed, all events corresponding to a single gate or net within the same frame are accumulated into the toggle rate for that frame. Reducing the number of frames over a set of events results in reducing the number of times a power model will be computed. This observation encourages us to investigate the impact of downsampling on the quality of side-channel leakage assessment. Clearly, power traces with fewer samples provide a simulation time advantage; the question is if they are still useful to identify side-channel leakage.

### 3.3 Sampling Power Traces

In this section, we discuss non-averaged and averaged sampling. Non-averaged sampling is prevalent in power measurement using oscilloscopes where each power sample is a snapshot of the power consumption in a moment of time. Averaged sampling is how power simulation tools produce traces where each sample is the averaged consumption in a moment of time. Averaged sampling is how power consumption of the design during the time interval of a frame is calculated. In the following, we give a mathematical model for each sampling type and study its implication on CPA as a measure of leakage assessment. A summary of the analyses is given in Table 2. Note that we do not perform averaging as an extra step in our proposed method. Rather, we start from the averaged traces as the output of commercial power simulators.

#### Table 2: Summary of sample types

| Sampling Type | Power Model | Implications for CPA |
|---------------|-------------|----------------------|
| 1) Non-averaged | Equation 1 | Theorem 3.1 |
| 2) Averaged - Constructive | Equation 3 | Theorem 3.4 |
| 3) Averaged - Destructive | Equation 4 | Theorem 3.6 |

#### 3.3.1 Non-Averaged Sampling

In non-averaged sampling, we use the following model for each power sample:

\[ P_{n,t_s} = \alpha \xi(x_n,k_{c_n}) + r_{n,t_s} + \mu_r \]  

(1)

where \( P_{n,t_s} \) denotes the power consumption at sample time \( t_s \) of the \( n^{th} \) power trace, \( \alpha \) is an unknown constant, \( \xi(x_n,k_{c_n}) \) is the selection function dependent on the controllable variable in trace \( n \) \((x_n)\) and the correct key in time sample \( t_s \). We model the algorithmic noise as a zero-mean random noise \( r_{n,t_s} \) and a constant unknown bias \( \mu_r \).

This model is similar to the model used by Fei et al. [5]. However, each trace may contain leakage of multiple key bytes, so that one trace can correlate with different correct key values. An example is the SubBytes function in the AES-128 algorithm, which will process 16 key bytes in every round.

#### Implications for CPA attack

Next, we derive the performance of a CPA attack on non-averaged traces.

**Theorem 3.1.** In non-averaged sampling, a sufficiently large number of traces is required for CPA attack to be successful.

**Proof.** The correlation coefficient between the selection function with key guess and the power trace is the covariance between the two sets divided by their standard deviations:

\[ \rho = \frac{\sum_{j=1}^{N} \left( k_j \cdot \xi_j \right) \left( P_{j,t_s} - \mathbb{E}_n[P_{n,t_s}] \right)}{\sqrt{\sum_{j=1}^{N} \left( \Delta \xi_j \right)^2} \sum_{j=1}^{N} \left( P_{j,t_s} - \mathbb{E}_n[P_{n,t_s}] \right)^2} \]  

(2)

where \( k_j \) is the key guess, and \( \Delta \xi_j \) is the correct key in time sample \( t_s \). The only term affected by the random noise in \( \rho \) is \( P_{j,t_s} - \mathbb{E}_n[P_{n,t_s}] \). Given enough constructive samples in a power trace and key value stronger by removing the random noise, corresponding to the same key value make the correlation between the two key values processed during the time interval, 2) multiple key values are processed during the time interval.

**Case 1** One key value during the time interval.

\[ P_{n,t_{s_1},t_{s_2}} = \mathbb{E}_t(t_{s_1},t_{s_2}) [P_{n,t_s}] \]

\[ = \alpha \mathbb{E}_t(t_{s_1},t_{s_2}) [\xi(x_n,k_{c_n})] + \mathbb{E}_t(t_{s_1},t_{s_2}) [r_{n,t_s}] + \mu_r. \]  

We separately analyze two scenarios in the following: 1) only one key value is processed during the time interval, 2) multiple key values are processed during the time interval.

**Theorem 3.2.** With enough time samples processing the same key value, the averaged power samples are shifted by the constant bias in the algorithmic noise.

**Proof.** For a sufficiently long interval, we have:

\[ \lim_{t_{s_1} \to \infty} \mathbb{E}_t(t_{s_1},t_{s_1} + \Delta t_{s_1}) [r_{n,t_s}] = 0. \]

In practice, given a long enough time interval correlating to one key value, i.e., \( \Delta t_{s_1} > t_{s_1} - t_{s_2} \), we have:

\[ P_{n,t_{s_1},t_{s_2}} \approx \alpha \xi(x_n,k_{B_n}) + \mu_r. \]

**Definition 3.3 (constructive samples).** Since power samples corresponding to the same key value make the correlation between the power trace and key value stronger by removing the random noise, we name them constructive samples.

**Implications for CPA attack.** Assuming averaged sampling of constructive samples, the following theorem holds.

**Theorem 3.4.** Given enough constructive samples in a power trace by design, averaged sampling will require fewer traces for CPA to succeed than non-averaged sampling.

**Proof.** The correlation coefficient between the selection function with key guess and the power trace is the same as Equation 2 replacing the power samples with Equation 3. Given enough constructive samples to be averaged in the interval \((t_{s_1},t_{s_2})\), we have:

\[ P_{j,t_{s_1},t_{s_2}} - \mathbb{E}_n[P_{n,t_{s_1},t_{s_2}}] = \alpha \left( \xi_j - \mathbb{E}_n[\xi_j(x_n,k_{B_n})] \right). \]
Therefore, the correlation is not affected by the algorithmic noise and CPA can converge with fewer traces.

However, without enough constructive samples, we have:

\[ P_{j,t_1,t_2} = E_n[ P_{n,t_1,t_2} ] = \alpha \cdot \zeta(x_j, k_B) + \varepsilon_n \left( \xi(x_n, k_B') \right) \]

Therefore a sufficiently large number of traces is required for the random noise to reach its zero mean and CPA successful. □

Case 2) Multiple key values during the time interval. We assume key value \( k_B \) is processed in the first part of the interval \((t_{s1}, t_{s2})\) and other key values from the set \( \{ k_B', k_B'' \} \) are processed in the rest of the interval \((t_{s3}, t_{s4})\). To further simplify the analysis, we assume the time samples in interval \((t_{s1}, t_{s2})\) correspond to processing the same key value \( k_B' \neq k_B \). The power sample in interval \((t_{s1}, t_{s2})\) can be written as:

\[ P_{n,t_1,t_2} = \alpha \cdot \zeta(x_n, k_B) + \varepsilon_n \left( \xi(x_n, k_B') \right) \]

Definition 3.5 (destructive samples). Since the power samples corresponding to different key values being processed makes the correlation between the power trace and the secret key value weaker, we name them destructive samples.

Implications for CPA attack. Similarly, Theorem 3.6 is derived for CPA attack.

Theorem 3.6. Averaging destructive samples diminishes the success probability of CPA attack even with a large number of power traces.

Proof. The correlation coefficient between the selection function and the power trace is the same as the previous case. Similarly, following the simplified averaged sample in Equation 4, we have:

\[ P_{j,t_1,t_2} = \alpha \cdot \zeta(x_j, k_B) + \alpha \cdot \zeta(x_n, k_B') + \varepsilon_n \left( \xi(x_n, k_B') \right) \]

where \( \Delta \zeta = \zeta(x_j, k_B) - \varepsilon_n \left( \xi(x_n, k_B) \right) \). Therefore, even with enough traces to cancel out the random noise, the term \( \alpha \cdot \Delta \zeta \) greatly impairs the correlation.

3.4 Empirical verification of theorems

Setup. Following the assumption that at each clock trigger there is a rush of current in the circuit, we generate power traces in the shape of sawtooth function where the amplitude is a function of the current level and the period is similar to the system clock period. This model follows the so-called Delta-I noise [6] and is similar to the model used by Tiran et al. [15]. SCA attacks, like DPA and CPA, exploit the dependency of the power consumption of a circuit on the secret data as well as a controlled data. To make our generated power traces applicable to such attacks, we further make the amplitude of the samples in each trace dependent on a function of a secret and a controlled byte. For our traces, we choose \( f_n(t) = x_n \cdot k_B \) where \( x_n \) is the controlled value for the \( n^{th} \) trace and \( k_B \) is the secret value for the time sample \( t \). Each trace corresponds to one controlled value and contains 64 clock cycles. This is similar to measured traces in practice where an attacker feeds a device with known inputs and acquires one trace for each given input. Every 16 consecutive clock cycles correspond to one key byte value (constructive). These traces are interpolated in Matlab up to a continuous-time power trace such that we can create both a non-averaged sample trace (oscilloscope) as well as an averaged sample trace (CAD power simulation) from the same source version. We generate 256 such traces, one for each controlled byte value.

Experiment 1 - averaged vs. non-averaged sampling. For each of the 256 traces, we generate noisy continuous-time traces with Signal-to-Algorithmic Noise Ratio (SANR) values of \{0, 5, 1, 0.5, 0.25, 0.1\}. For each SANR value we create two power traces from the same source data: a non-averaged sampled version with two samples per clock cycle, and an averaged sampled version with two frames per clock cycle. We run CPA on the two sets of noisy traces for each key byte separately and calculate the rank of the correct key value. We repeat the same process of noisy trace generation, sampling, and CPA 100 times and report the average correct key rank and average success rate (SR) calculated as SR = total number of attacks / (total number of attacks + number of successful attacks) in Figure 2.

This experiment shows a substantial improvement of averaged traces over non-averaged traces. For example, attacks on averaged traces approach a success rate of 1 as soon as SANR > 0.5, while non-averaged traces achieve the same success rate only at SANR > 10.

Experiment 2 - constructive samples. For the same set of power traces with SANR=0.1 (weakest SANR), we average up to 16 clock cycles corresponding to the same key byte (constructive samples). We report the SR of CPA attack to compare the different levels of constructive averaging in Table 3. As this table shows, as we average more constructive samples, it becomes more likely to find the correct key.

Table 3: SR of CPA on key bytes in simulated traces for different number of averaged constructive samples with SANR= 0.1 (averaged over 100 runs).

| # Constructive | SR byte 1 | SR byte 2 | SR byte 3 | SR byte 4 |
|---------------|-----------|-----------|-----------|-----------|
| 2             | 0.68      | 0.61      | 0.68      | 0.66      |
| 4             | 0.78      | 0.78      | 0.79      | 0.71      |
| 8             | 0.83      | 0.92      | 0.83      | 0.81      |
| 16            | 0.92      | 0.96      | 0.9      | 0.95      |

Experiment 3 - destructive samples. We average 16 constructive cycles for each key byte and add up to 16 destructive cycles to the averaged sample to study the effect of destructive samples on CPA results. Table 4 shows the SR decreases as more destructive samples are added, attesting to Theorem 3.6.

4 CASE STUDIES

In the following case studies, we show how different hardware designs affect the level of averaging that can be done without loss.

1We differentiate between the SNR pertaining to the measurement noise and the SNR pertaining to the algorithmic noise, calling the latter Signal-to-Algorithmic Noise Ratio (SANR). Algorithmic noise is any signal not correlated with the target of the attack.
of precision. Our case studies contain a pipelined microprocessor running software AES, and an AES hardware accelerator. We chose AES-128 as its vulnerabilities are well-understood in the literature, and therefore this case makes a good driver for our experiments.

### 4.1 Case Study 1: Software AES on a Pipelined Processor

For our pipelined processor, we use the five-stage BRISC-V\(^2\) implementation of RISC-V RV32I ISA (integrated into a system-on-chip [9]). The five stages in this core are instruction fetch, instruction decode and operand access, execution, memory access, and write back. We synthesize the design for the open standard cell library SkyWater 130nm and 50MHz frequency using Cadence Genus. In each gate-level simulation, we load the binary file of the compiled AES software code into the program memory of the processor, feed AES-128 as its vulnerabilities are well-understood in the literature, (Table 4: SR of CPA on key bytes in simulated traces for different ratio of destructive to constructive samples with SANR=0.1 (averaged over 100 runs)).

![Figure 3: Flow of instructions in Listing 1 through five stages of RISC-V pipeline. Highlighted cells show the leaking parts.](https://ascslab.org/research/briscv/index.html)

### Listing 1: Assembly code for SubBytes on RISC-V

|   | Clock Cycle | Fetch | Decode | Execute | Memory | Write Back |
|---|-------------|-------|--------|---------|--------|------------|
| 1 | F11         | D10   | -      |         | W9     |            |
| 2 | F12         | D11   | E10    |         |        |            |
| 3 | stall       | stall | M10    | flush   |        |            |
| 4 | stall       | stall | stall  |         |        |            |
| 5 | F12         | D11   | stall  | E11     | W10    |            |
| 6 | F13         | D12   | E11    | stall   |        |            |
| 7 | F14         | D13   | E12    | M11     | stall  |            |
| 8 | stall       | stall | stall  | flush   | M12    | W11        |

![Figure 4: CPA ρ vs. sample number for locating the power leakage of last key byte (B15) from coarse (1s50cc) to finer (1s1cc) traces for software AES running on RISC-V. Grey lines are correlation values for incorrect key guesses and black line is the correlation value for correct key guess.](https://ascslab.org/research/briscv/index.html)

### Table 5: Speed-up of averaged RISC-V traces compared to 1s1cc

| Num. of Averaged Clock Cycles | 1s10cc | 1s20cc | 1s30cc | 1s40cc | 1s50cc |
|------------------------------|--------|--------|--------|--------|--------|
| Simulation Speed-up          | 3.35X  | 4.86X  | 5.45X  | 5.78X  | 6.50X  |
4.2 Case Study 2: Hardware AES

In the next experiment, we shift our focus from constructive samples to destructive samples. We consider a hardware accelerator for AES-128 that runs 4 SBoxes in parallel in each clock cycle, therefore, destructive samples are intrinsically present in each power sample. Figure 5 shows the structure of the AES implementation. Each AES round is executed in 5 clock cycles. The first four clock cycles each execute 4 SBoxes (grey shade in Figure 5). The \( ma \) (resp. \( mb \)) multiplexers choose which 32bit part of the state (SW0 through SW3) should be updated (resp. calculated) through the SBox modules. The last clock cycle executes the rest of the blocks in succession (SR, MC, and AddRoundKey) following which \( ma \) multiplexers choose the appropriate update values for each state word. \( mc \) sets the correct input for the AddRoundKey step, i.e., plaintext (\( P \)) at the start, MC output for the first 9 rounds, and SR output for the last round.

We synthesize the AES module for SkyWater 130nm standard cell library and 50MHz clock frequency using Cadence Joules and black line is the correlation value for correct key guess. The last clock cycle executes the rest of the blocks in succession (SR, MC, and AddRoundKey) following which \( ma \) multiplexers choose the appropriate update values for each state word. \( mc \) sets the correct input for the AddRoundKey step, i.e., plaintext (\( P \)) at the start, MC output for the first 9 rounds, and SR output for the last round. We synthesize the AES module for SkyWater 130nm standard cell library and 50MHz clock frequency using Cadence Joules.

In the first round of AES, each SW register will overwrite the first AddRoundKey output with its corresponding SBox output. Therefore, for CPA, we use HD(\( k_p,p \), SBox(\( k_p,p \))) as the leakage model. Figure 6 shows the CPA correlation result for the last subkey as an example. The correlation values have drastically reduced and in some cases (e.g. \( 1s4cc \) for subkey 15) the CPA fails. This example shows that a secure hardware designer should avoid combining destructive samples in the power analysis stage to prevent false negative conclusions in the assessments.

Discussion. As shown experimentally and theoretically, a secure hardware designer can take advantage of constructive samples in a design to downsample traces and reduce the power simulation time which is the bottleneck for pre-silicon PSCL assessment. At the same time, the designer should be aware that averaging destructive samples can lead to false negative conclusions in leakage assessment. Fortunately, classifying power samples as constructive or destructive is straightforward for well-known algorithms and implementations such as the ones analyzed in our case studies. However, additional preprocessing may be needed for unknown designs. For instance, using specific test vectors that only exercise part of the key and applying statistical tests to tag a given power sample as destructive or constructive. In our future work, we study techniques that can help a designer classify power samples as such for certain leakage criterion.

5 CONCLUSION

We studied the implication of averaged sampling for pre-silicon side-channel leakage assessment. We introduced the concept of constructive and destructive samples as categories of power samples which will facilitate or hinder the leakage assessment of a certain sensitive data. We showed how using downsampling in designs for which the constructive samples are well understood aid in reducing power simulation time without significant loss in precision. In future work, we will study tests that can categorize samples as constructive or destructive to be applied to less-known designs.

ACKNOWLEDGMENTS

This research was supported in part by NSF award CNS-1931639.

REFERENCES

[1] G. T. Becker et al. 2013. Test Vector Leakage Assessment (TVLA) methodology in practice. In International Cryptographic Module Conference.

[2] E. Brier et al. 2004. Correlation power analysis with a leakage model. In International workshop on cryptographic hardware and embedded systems. Springer, 16–29.

[3] I. Buchan et al. 2021. SoK: Design Tools for Side-Channel-Aware Implementations. arXiv preprint arXiv:2104.08593 (2021).

[4] T. El Motassadeq. 2011. CCS vs NLD comparison based on a complete automated correlation flow between PrimeTime and HSPICE. In 2011 Saudi International Electronics, Communications and Photonics Conference (SIECPC). IEEE, 1–5.

[5] Y. Fei et al. 2014. A Statistics-based Fundamental Model for Side-channel Attack Analysis. IACR Cryptol. ePrint Arch. 2014 (2014). 152.

[6] J. L. Gonzalez et al. 1999. Low delta-I noise CMOS circuits based on differential logic and current limiters. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications 46, 7 (1999), 872–876.

[7] M. He et al. 2019. RTL-FSC: Automated power side-channel leakage assessment at register-transfer level. In 2019 IEEE 37th VLSI Test Symposium (VTS); IEEE, 1–6.

[8] M. A. KF et al. 2020. PARAM: A microprocessor hardened for power side-channel attack resistance. In 2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). IEEE, 23–34.

[9] P. Kiaeri et al. 2021. Architecture Support for Bitlicing. Cryptology ePrint Archive, Report 2021/1236. https://ia.cr/2021/1236.

[10] P. Kiaeri et al. 2021. Saidoyoki: Evaluating side-channel leakage in pre-and-post silicon setting. IEEE International System-on-Chip Conference (SOCC) (2021).

[11] P. Kocher et al. 1999. Differential power analysis. In Annual international cryptology conference. Springer, 398–397.

[12] D. Šijačić et al. 2020. Towards efficient and automated side-channel evaluations at design time. Journal of Cryptographic Engineering 10, 4 (2020), 305–319.

[13] P. Slpšk et al. 2019. Karna: A gate-sizing based security aware eda flow for improved power side-channel attack protection. In 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). IEEE, 1–8.

[14] F.-X. Standaert. 2018. How (not) to use welch’s t-test in side-channel security evaluations. In International Conference on Smart Card Research and Advanced Applications. Springer, 65–79.

[15] S. Tiran et al. 2014. A model of the leakage in the frequency domain and its application to CPA and DPA. Journal of Cryptographic Engineering 4, 3 (2014), 197–212.

[16] Y. Yao et al. 2020. Architecture correlation analysis (ACA): identifying the source of side-channel leakage at gate-level. In 2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). IEEE, 188–196.

[17] Y. Yao et al. 2020. Verification of Power-based Side-channel Leakage through Simulation. In 2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS). 1112–1115.