Root-cause Analysis of Power-based Side-channel Leakage in Lightweight Cryptography Candidates

Zhenyuan Liu and Patrick Schaumont[0000−0002−4586−5476]

Worcester Polytechnic Institute, Worcester MA 01609, USA

Abstract. We present a detailed analysis of the root cause of power-based side-channel leakage in candidates of the NIST lightweight cryptography competition. We perform gate-level power simulation, and rank the contribution of individual cells to the overall side-channel leakage. The proposed leaky-gate selection proceeds in two steps. For a selected set of test vector stimuli, we first identify *leaky points*, which indicate the time stamps of maximum data-dependent variation in the power traces. Next, we rank the side-channel leakage of each individual cell according to their power-based standard deviation at the selected leaky points. We analyze the distribution of side-channel leakage over different cell types. We highlight the root causes of side-channel leakage at the gate level for selected NIST lightweight cryptographic candidates including a block cipher (GIFT-COFB), a sponge-based cipher (Xoodyak) and a stream cipher (Grain-128). We compare these findings to a traditional AES implementation.

Keywords: Power-based Side-channel Leakage · Gate-level Simulation · Pre-silicon Side-channel Leakage Analysis

1 Introduction

Pre-silicon side-channel leakage assessment refers to the use of electronic design automation to determine the side-channel leakage properties of a hardware design before manufacturing. In the past few years, this area has enjoyed significant interest and a wealth of tools and techniques have been proposed to tackle the challenge [BBYS21]. A straightforward application of pre-silicon side-channel leakage assessment is to demonstrate the presence of side-channel leakage from a design description. Pre-silicon side-channel leakage assessment requires the simulation of a cipher's implementation in terms of power (for power-based side-channel leakage) and/or execution time (for timing-based side-channel leakage). Next, the estimated power or timing are evaluated for dependencies on internal secret values. A broad range of metrics have been proposed in recent years and a comprehensive overview is presented by Papagiannopoulos *et al.* [PGA+22].

In this contribution, we go beyond side channel leakage assessment and seek to highlight the *root-cause* of side-channel leakage. The root-cause analysis leads

to the identification of the cells that cause data-dependent power variation [1]. For a given hardware design, we propose a ranking of the logic cells in the design according to their contribution to data-dependent power consumption. We use a non-specific leakage criteria[2]. Our ranking is directly driven by statistical properties of the logic-cell power consumption, and it is independent of high-level power models. This allows the comparison of cryptographic *implementations*. Indeed, the NIST Light Weight Cryptographic Competition has selected candidate primitives with varying compositions including permutations, block ciphers, tweakable block ciphers, and stream ciphers. We aim to compare the side-channel leakage of their implementation in a manner independent of the power leakage model used for a concrete side-channel attack.

Our objective is to analyze unprotected implementations, and to understand how the different types of logic cells that make up a hardware cipher contribute to the overall power-based side-channel leakage of a cipher. We do not aim at demonstrating that a (possibly protected) hardware cipher does *not* leak. Strictly speaking, pre-silicon side-channel leakage assessment is unable to confirm the *absence* of side-channel leakage. Although specific properties such as the distribution of mask values can be formally verified [WS17], simulations are always limited in accuracy by their modeling capabilities, and therefore they do not represent the physical world with full accuracy. There are a number of well-known examples of how high-level models can hide side-channel leakage at lower abstraction levels. For example, cycle-accurate simulation cannot capture sub-cycle effects such as glitches [NRR06]. For example, gate-level simulation does not capture capacitive coupling effects introduced by physical placement and routing [DCBG+17].

This paper uses the following organization. In the next Section, we develop the methodology for root cause analysis and apply it to the design of a hardware AES encryption/decryption module. In Section 3, we apply the methodology to each of three selected Lightweight Cryptography candidates including GIFT-COFB, Xoodyak, Grain128-AEAD. For each of these ciphers, we determine a leaky cell ranking, and demonstrate the sensitivity of the side-channel leakage of the overall cipher to individual leaky cells. In Section 4, we analyze the implementation complexity of each cipher, as well as the distribution of leaky-cell types over the implementations.

2 Root Cause Analysis Methodology

In this section, we explain our root cause analysis methodology. The objective of this analysis is to identify those logic cells in a design that are the biggest contributor to side-channel leakage. In the context of this paper, side-channel

[1] We use the term *gate-level* to express the abstraction level that uses logic cells, registers, etc. We use the term *cell* to describe an instance of a standard-cell library.

[2] Non-specific leakage criteria avoid making assumptions how side-channel leakage is used in a side-channel attack.

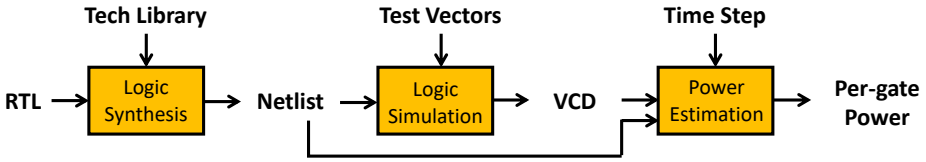

Fig. 1. Per-gate power estimation

leakage specifically refers to *a power consumption dependency on a controlled but secret internal value.*

Power Simulation We use gate-level power simulation to obtain high-resolution per-gate power consumption traces as shown in Figure 1. A design at register-transfer level is synthesized to a netlist using a standard cell library. The resulting netlist is simulated under selected test vectors, while recording the activity of every cell in the design. The activity is stored in a *Value Change Dump* file. The VCD file and the netlist are then used in a power estimation tool which will compute a power trace at the resolution of a selected time-step. In our root-cause analysis methodology, we compute per-gate power traces. The power consumption of a design is therefore characterized in a three-dimensional structure as $P(v, t, g)$, with v the test-vector, t the discrete-time stamp in a power trace, and g the cell in the design. For the experiments in this contribution, we use Cadence Genus 20.10-p001_1 for logic synthesis and Cadence Joules v20.11-s001_1 for gate-level power simulation. We target 130 nm technology standard gates of the SkyWater Open Source Technology Library.

Selecting Test Vectors Next we build a methodology to rank the cells in a design according to their contribution to the side-channel leakage. We aim to have a ranking criteria that is independent of a high-level power-model. First, we show that by careful selection of the input test vectors and of the range of time stamps covered by the simulation, we are able to maximize the proportion of side-channel leakage in the overall power consumption of a design. Next, we describe a two-step iterative process to select and rank cells according to side-channel leakage.

Consider the hypothetical cipher in Figure 2 which performs an encryption. The side channel leakage of the design is measured as the combined power consumption of every logic cell within the dashed box. The logic cells in the cipher use a plaintext input and a secret key, but only the plaintext input can be externally controlled. All cells that depend on a primary input are said to be included in the *logic cone* of that input. We can thus think of a logic cone driven from the plaintext input, and a second cone driven from the secret key. This partitions the logic cells into one of four groups: logic cells contained exclusively in the logic cone of the plaintext input, logic cells contained exclusively in the logic cone of the secret key, logic cells contained in the logic cone of both the plaintext input as well as the secret key, and finally logic cells outside of any logic cone.

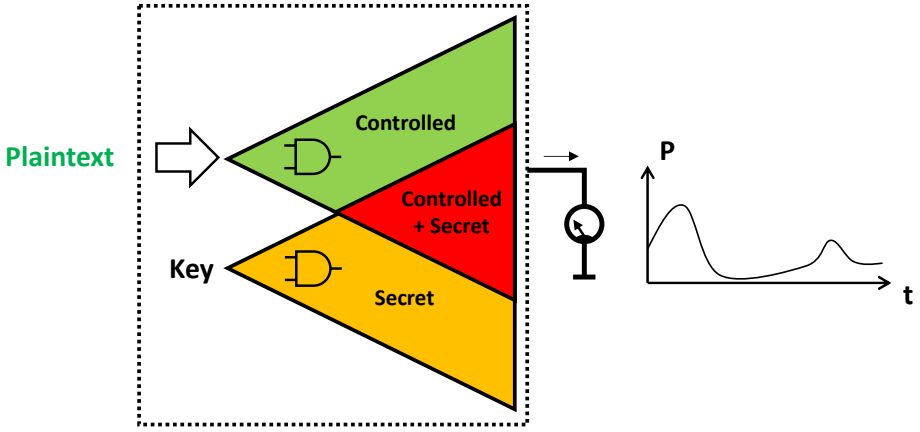

Fig. 2. Three kinds of data-dependent power consumption

The most popular form of side-channel attack, differential power analysis, targets the cells contained in the red logic cone shared between the secret key and the (plaintext) input. Cells outside of any logic cone, as well as cells in the green logic cone, are not useful for side-channel attacks. Cells in the orange logic cone are only dependent on the secret key. These cells *may* be contributing to a side-channel attack based on simple power analysis or a template. This is generally considered a harder attack because only a single trace is available. We therefore focus our analysis on the case of differential power analysis.

By carefully selecting test input vector values, as well as the time range of the power simulation, we aim to maximize the activity in the overall design to cells located in the red cone. For example, in case of the Advanced Encryption Standard, we can use a random plaintext input and limit the power trace time span to the key pre-whitening and first-round substitution phase. This ensures that all cipher activity recorded in the VCD trace represents the interaction of a controlled input and a secret, and therefore that it will include cells that are contained in the red logic cone of Figure 2.

Macro-level Analysis After selection of the test vectors and simulation time range, we obtain the detailed power traces for individual cells. Intuitively, the cells that cause the largest variation in power consumption are also those that are the main contributors to side-channel leakage. However, there is an important caveat: a cell may cause leakage over multiple clock cycles. Hence, we may need to combine the power variation of a cell over multiple time points, in order to correctly reflect all the side channel leakage coming from a cell. This problem, in turn, leads to another challenge, namely how to select time points at which a cell is considered 'leaky'. We solve this problem with a macro-level analysis of the power traces for the entire design. We select those time points in the overall power trace that show a maximum variation for the selected test vectors and

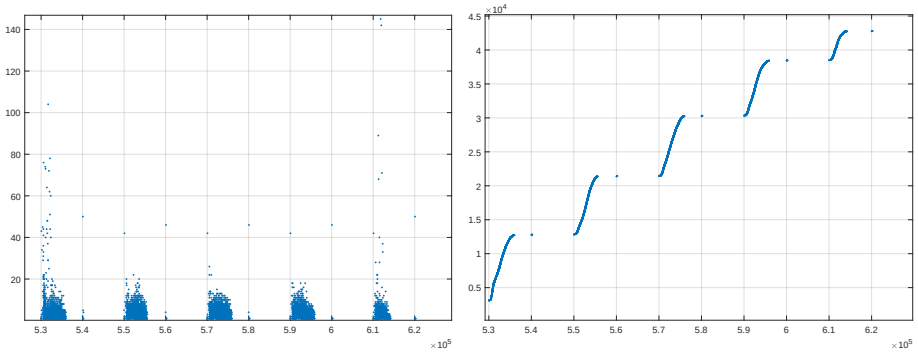

Fig. 3. (Left) Event Density Plot for 5 cycles (one round) of an AES standard cell design (Right) Cumulative Event Density Plot for 5 cycles (one round) of an AES standard cell design

simulation time range. The rationale of this choice is the following. Side-channel leakage is largest at time points with the highest data-dependency of the power consumption, in other words, at points of maximum variation in the power trace. Any cells that can be identified as switching at or near those local maxima, can be called leaky cells, because it is their combined activity that causes the large peak in power variation in the first place.

Before elaborating the macro-level analysis, we demonstrate that the design-global power trace is typically orders of magnitude larger than the per-cell power trace. Figure 3 (Left) shows an event density plot of 5 clock cycles of an AES design mapped to standard cells. The X axis represents time in picoseconds, and the Y axis represents the number of events occuring at a specific time stamp. The five clock cycles represent activities in the first round of the design - namely when the master key is combined with the plaintext input. Each point of the scatterplot represents a logic transition at the output of a cell, and there are almost 45,000 events over the five clock cycles. Each clock cycle consists of two portions. Following an up-going clock edge (e.g., at 530 ns, 550 ns, 570 ns, etc.), the register outputs change and a large portion of the combinational cells switch to evaluate a new result. Next, at the down-going clock edge (e.g. at 540 ns, 560 ns, 580 ns, etc), a much smaller portion of cells are updated, as the register outputs do not change. The event density plot shows that up to 100 cells can switch at the same time stamp. Within a clock cycle, the combinational cells switch over a time span of almost 5 ns, which is a result of multiple layers of logic in the implementation. However, the overall number of switching events in a clock cycle is enormous. Figure 3 (Right) shows a cumulative event density plot: time point x shows the total number of events with time stamp smaller than x. The x axis represents time in picoseconds, the y axis represents events (in 500K units). Every clock cycle adds around 11,000 events, implying that almost 11,000 logic transitions can occur in a single clock cycle in this particular AES design.

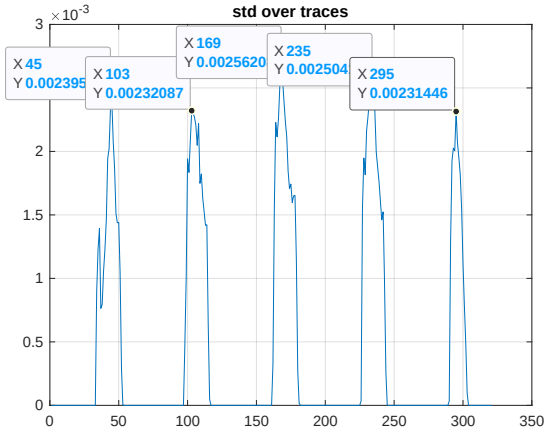

Fig. 4. Standard Deviation of Power over 256 power traces of a first-round AES implementation

We can now determine the set of leakage time points as follows. Obtain the design-global power trace $P(v,t)$ by accumulating the per-cell power traces. Compute $s(t)$, the standard deviation as a function of time. Finally, determine a set of local maxima in the standard deviation. The set becomes the leaky point set $T = \{t_1, t_2, t_3, ...\}$. The number of points selected to be part of the leaky point set is flexible, but typically we select a single point per simulated cycle in the time range under consideration (such as one round of a cipher).

$$P(v,t) = \sum_{g} P(v,t,g) \tag{1}$$

$$s(t) = \underset{v}{std}(P(v,t)) \tag{2}$$

$$T = \{t_1, t_2, t_3, ...\} = \underset{t}{localmax}(s(t)) \tag{3}$$

This leads to the first insight in leaky cell selection: a side-channel analysis will be concentrating on points of maximal power variation *at the macroscopic level*. We are therefore looking for cells that switch at the time instant of maximal power variation in the power trace.

Micro-level Analysis After selection of the macro-level leakage points, we can now select a ranking criteria for individual cells. For each cell g and each test vector v, we create a leakage estimation $leakage(v,g)$ by adding the power consumed by each individual cell at each selected macro-level leakage points. Next, we compute the standard deviation of the estimated leakage $l(g)$ for each cell over the selected test vectors. Finally, we rank the cells according to their leakage $l(g)$.

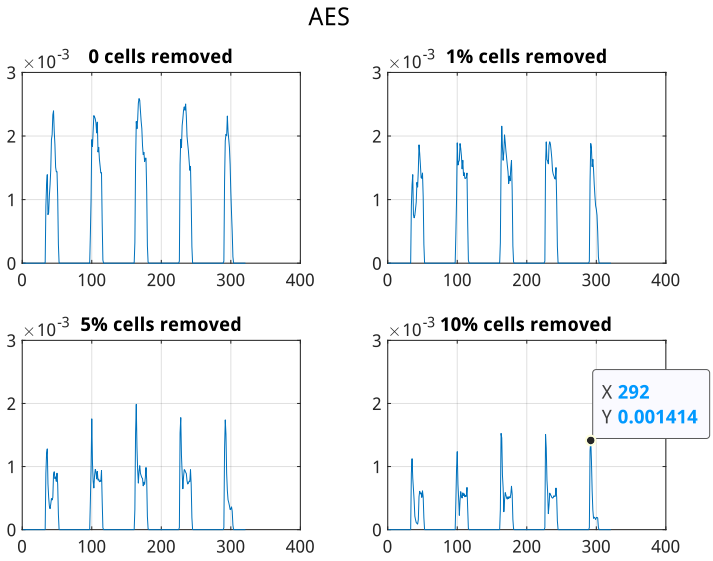

Fig. 5. Impact on Power Standard Deviation after removing 1%, 5% and 10% of the cells

$$leakage(v, g) = \sum_{T} P(v, t, g) \tag{4}$$

$$l(g) = \underset{v}{std}(leakage(v, g)) \tag{5}$$

$$G = \{g_1, g_2, g_3, ...\} = argrank(l(g)) \tag{6}$$

Figure 4 shows the standard deviation of the power consumption of the first round of an AES hardware implementation. This 9,640-cell design has a 32-bit datapath and computes each round over 5 clock cycles. The first four clock cycles of the round evaluate the SBOX-lookups, while the last clock cycle evaluate the other steps of the round. We apply 256 test vectors consisting of a constant-key and a random plaintext. The figure shows a peak at each upgoing clock edge. There is no peak at the downgoing edge indicating that power associated to the downedge events show no variation with the plaintext. To select the leakage model, we selected one peak per clock cycle at macro-level. In Figure 4, these are marked at point 45, 103, 169, 235 and 295. The estimated leakage for each cell is the sum of the power of each cell at these selected time points. From the estimated leakage, the cell ranking can be computed according to the formulae above.

To evaluate the quality of the ranking, we demonstrate that the standard deviation of the power will decrease by removing top-ranked cells. Figure 5 shows

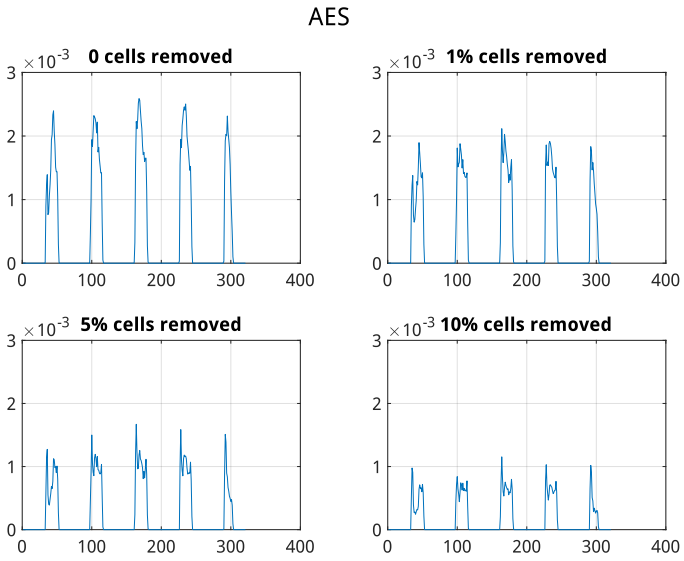

Fig. 6. Second Iteration Impact on Power Standard Deviation after removing 1%, 5% and 10% of the cells

the resulting standard deviation on power traces after the top 1%, 5%, 10% of cells are removed. There is a drastic reduction of power variation after removing even a minor portion of cells, indicating that the cell selection methodology is able to identify those cells that cause the largest power variations (and, by inference, the largest amount of side-channel leakage).

We notice that after the first iteration of removing top-ranked cells, some residual leakage peaks are remaining at each upgoing clock edge on the power variation. These peaks are at position 35, 100, 163, 227, and 292 in Figure 5. We improved the leakage model by adding these residual peaks to the first iteration leakage model. In this case, we have selected two peaks per clock cycle for the leakage model at the macro-level. Figure 6 shows the improved resulting standard deviation on power traces after the top 1%, 5%, 10% of cells are removed. The resulting standard deviation from the second iteration is more uniformly distributed compared to the first iteration.

To further validate if those top-ranked logic cells in a design are the biggest contributor to side-channel leakage, we perform Correlation Power Analysis (CPA) on AES. Figure 7 shows an example of a byte-wise CPA on AES before and after removing an increasing portion of the top-ranked cells. The gradual increase of the number of measurements before CPA success indicates that the algorithm successfully identified leaky cells.

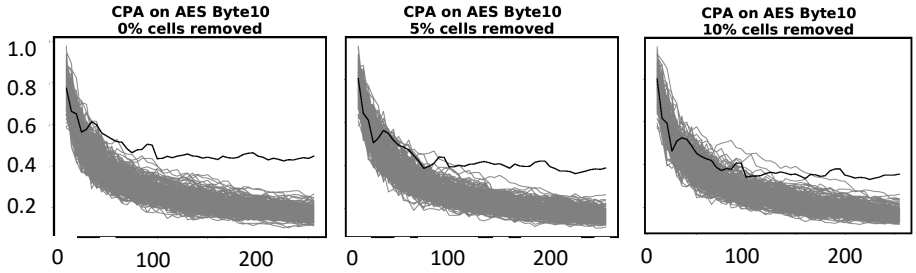

Fig. 7. CPA on AES byte 10 with cells being removed according to ranking shows a gradual increase of the number of traces needed for disclosure

3 Root Cause Analysis of Selected Lightweight Cryptography Candidates

In this section, we explain our root cause analysis methodology on several lightweight cryptography candidates, including a block cipher (GIFT-COFB), a sponge-based cipher (Xoodyak), and a stream cipher (Grain128-AEAD).

Block Cipher GIFT-COFB instantiates the COFB (COmbined FeedBack) block cipher based Authenticated Encryption with Associated Data (AEAD) mode with the GIFT-128 block cipher [BCI+20]. GIFT-COFB encryption inputs a 128-bit encryption key, a 128-bit nonce, an associated data and a message of arbitrary length, and outputs a ciphertext of the same length as the message, and a 128-bit tag. GIFT-128 is a 40-round 128-bit Substitution-Permutation network (SPN) based block cipher. We use an open-source GIFT-COFB hardware implementation [3]. The entire encryption of the nonce takes 40 clock cycles to finish. Each block of the associated data or plaintext takes 40 cycles to process and 4 extra cycles to update the *delta* register.

To maximize the proportion of side-channel leakage in the overall power consumption of a design, we apply 256 test vectors consisting of a constant-key and a random nonce with neither associated data nor plaintext. Figure 8, top, shows the first 20 cycles of hardware GIFT-COFB. The top figure shows the power trace on one test vector, and the bottom figure shows the standard deviation of power over 256 power traces. This 3,286-cell design takes the first 20 cycles to process the 128-bit nonce, and the last 20 cycles to process the hard-coded 128-bit associated data when neither external associated data nor plaintext is loaded. The clock frequency is 50MHz, and the power simulation is oversampled at 50 samples per clock cycle. Figure 8, bottom, shows a peak at each upgoing clock edge except the first cycle. This indicates that there is no variation on the first cycle. To implement the leakage model, we select one peak per clock cycle at the macro-level. These peaks are marked at point 53, 104,

[3] https://github.com/qantik/Energy-Analysis-of-Lightweight-AEAD-Circuit/tree/master/gift-cofb/round

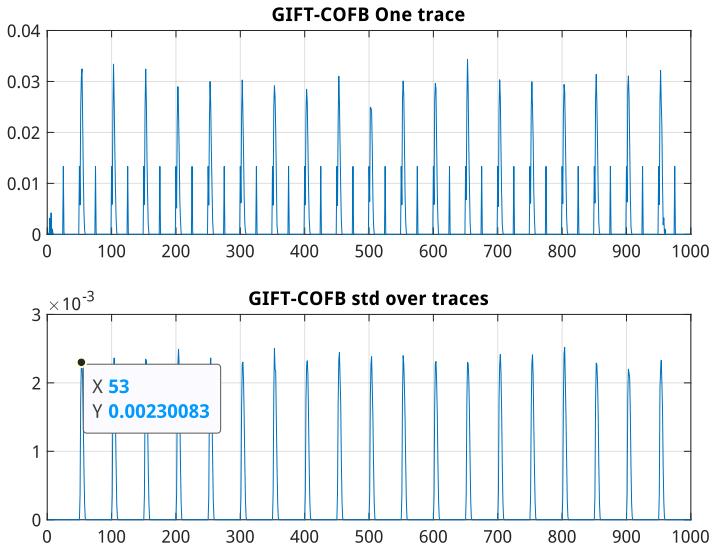

Fig. 8. (Top) First 20-round of GIFT-COFB Power trace on one test vector (Bottom) Standard Deviation of Power over 256 power traces of the first 20-round GIFT-COFB implementation

153, 204 and so on. The estimated leakage and the cell ranking are computed using the formula listed in Section 2. Perform two iterations of our cell-ranking algorithm, and end up with a power variation as shown in Figure 9. We see a drastic reduction after a minor portion of cells is removed. This proves the standard deviation of the power will decrease by removing top-ranked cells, and the cell selection methodology is able to identify those cells that cause the largest power variations.

Sponge-Based Cipher Xoodyak is an authenticated encryption and hash algorithm pair based around the 384-bit Xoodoo permutation [DHP+20]. Xoodyak encryption inputs a 128-bit key, a 128-bit nonce, and outputs a 128-bit authentication tag. The mode of operation on top of Xoodoo is called *Cyclist*, it converts the permutation into a sponge for the higher-level algorithms. Figure 10 shows the power of a full run of Xoodyak. We use an open-source hardware implementation of Xoodyak [4]. This implementation does not include hash functionality. This 3,630-cell design spends 4 rounds to store the encryption key, 4 rounds to absorb the nonce, 12 rounds to absorb the associated data, 12 rounds to pad the message, and 12 rounds to extract the tag. Each round takes 2 cycles to finish. There is one round of padding between each phase with smaller power peaks.

[4] https://github.com/KeccakTeam/Xoodoo/tree/master/Hardware/ASIC/AEAD/Xoodyak_R3

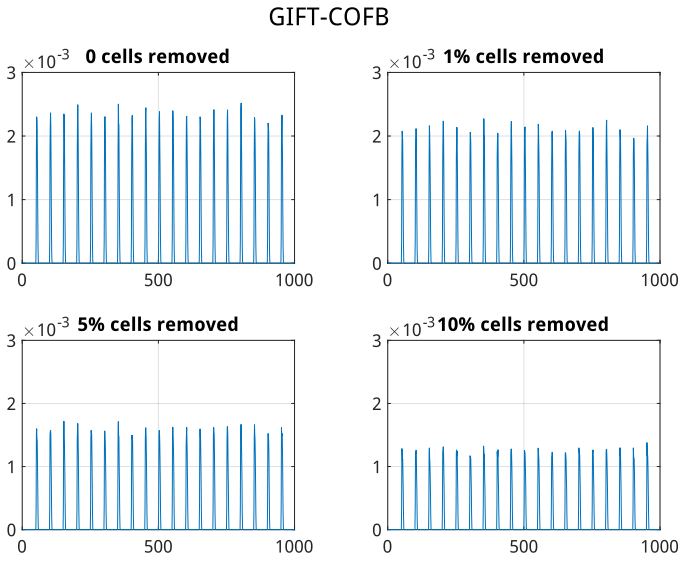

Fig. 9. Second Iteration Impact on Power Standard Deviation after removing 1%, 5% and 10% of the ranked cells in GIFT-COFB

We apply the same test vector selection methodology to Xoodyak. We simulate 256 test vectors consisting of a constant-key and a random nonce with neither associated data nor message. Figure 11, top, shows 8 cycles of the absorb nonce implementation. The top figure shows the power trace on one test vector. There are higher peaks at the first cycle of each round, and smaller peaks at the second cycle of each round. The bottom figure shows the standard deviation of power over 256 power traces. The second cycle of each round does not show variation associated with the nonce. The clock frequency is 50MHz, and the power simulation is oversampled at 64 samples per clock cycle. To select the leakage model, we selected one peak per round at the macro-level. These peaks are marked at point 69, 197, 327, and 454. The estimated leakage and the cell ranking are computed using the formula listed in Section 2.

Figure 12 shows the resulting standard deviation on power traces after the top 1%, 5%, 10% of cells are removed. There is a drastic reduction of power variation after removing a minor portion of cells. This validates the standard deviation of the power will decrease by removing top-ranked cells, and the cell selection methodology can identify those cells that cause the largest power variations.

Stream Cipher Grain128-AEAD is a stream cipher supporting authenticated encryption with associated data [SHSK19]. Grain128-AEAD encryption takes a 128-bit key, a 96-bit nonce, a plaintext of arbitrary length, an associated data of arbitrary length, and outputs a ciphertext of arbitrary length and a 64-bit

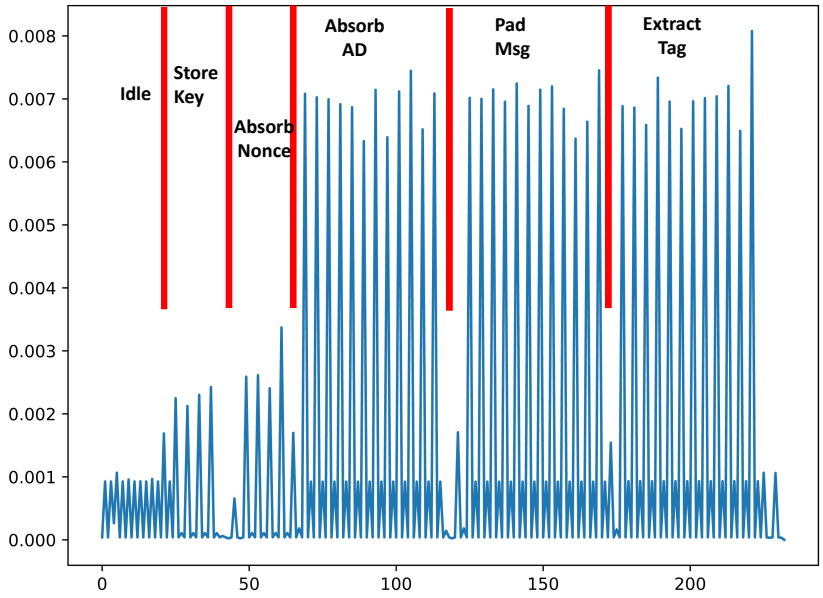

Fig. 10. Full View of Hardware Xoodyak Power Simulation

tag. We use an open-source hardware implementation of Grain128-AEAD [5]. It consists of two main building blocks. The first is a pre-output generator consisting of a Linear Feedback Shift Register (LFSR), a Non-linear Feedback Shift Register (NFSR), and a pre-output function. The second block is the authentication block consisting of a shift register and an accumulator. This 602-cell design divides the cipher into three stages. The first stage is the *loading stage*. It takes 128 cycles to load the key and the nonce. The last 32 bits of the nonce are filled with 31 ones and a zero. The second stage is the *initialization stage* which takes 256 cycles to finish. The last stage is the *running stage*, where the pre-output is generated for encryption and authentication.

We apply the same test vector selection methodology to Grain128-AEAD. We simulate 256 test vectors consisting of a constant-key and a random nonce with neither associated data nor plaintext. Figure 13 shows 256 cycles of the Grain128-AEAD during the initialization stage. The top figure shows the power trace on one test vector. The clock frequency is 50MHz, and the power simulation is sampled at 1 sample per clock cycle. The bottom figure shows the standard deviation of power over 256 power traces. To select the leakage model, we selected one peak for the entire 256 cycles at the macro-level. This peak is marked at point 205, which is the local maximum of the power standard deviation. The estimated leakage and the cell ranking are computed using the formula listed in Section 2.

[5] https://github.com/Grain-128AEAD/Grain-128AEAD-VHDL

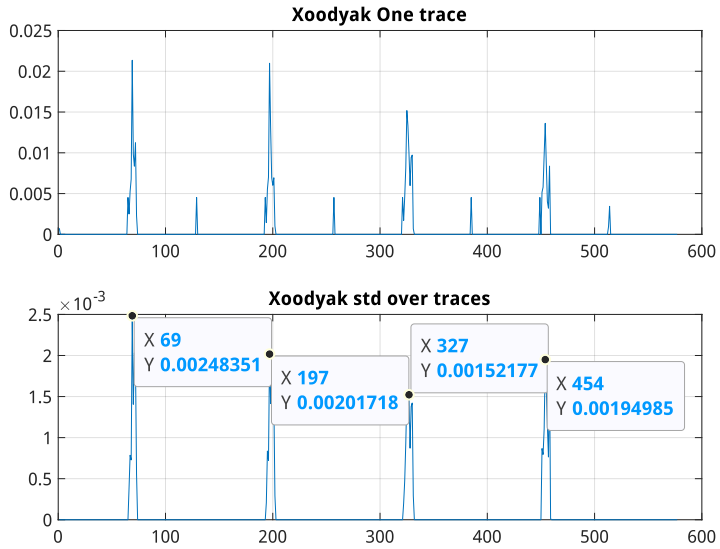

Fig. 11. (Top) Xoodyak Absorb Nonce Power trace on one test vector (Bottom) Standard Deviation of Power over 256 power traces of Xoodyak Absorb Nonce implementation

Figure 14 shows the resulting standard deviation on power traces after the top 1%, 5%, 10% of cells are removed. We observe a uniform reduction of power variation after removing a minor portion of cells with only one point selected in the leakage model.

4 Discussion

In this section, we present the comparison of the root cause analysis results among AES, GIFT-COFB, Xoodyak, and Grain128-AEAD in terms of cell count, cell type, area, and power consumption.

Table 1. Cell Count for each Cipher

Cipher	Total Cells	Combinational Cells	Sequential Cells
AES	9640	7121 (74%)	2519 (26%)
GIFT-COFB	3286	2955 (90%)	331 (10%)
Xoodyak	3603	3180 (88%)	423 (12%)
Grain128-AEAD	602	196 (33%)	406 (67%)

Fig. 12. Impact on Power Standard Deviation after removing 1%, 5% and 10% of the cells

Cell Count Table 1 shows the count number of total cells, combinational cells, and sequential cells in each cipher. We observe that in the block cipher and the sponge-based cipher, cell instances are dominated by combinational cells. Especially in GIFT-COFB, 90% of the cell instances are from the combinational cells. When the side-channel leakage is defined by a large number of combinational cells, it tends to be spread out over a clock cycle, as we could observe in the power plots for AES, GIFT-COFB, and Xoodyak.

On the other hand, sequential cells dominate the majority of the cells in the stream cipher. From Table 1, we observe that 67% of the cells are sequential cells in Grain128-AEAD. One of the properties in the stream cipher is that it performs bit-by-bit operations. This means that the flip-flops inside of the stream cipher switch at every up-going clock edge. This explains why with only one point selected in the leakage model, we are able to see a uniform reduction in the power variance of Grain128-AEAD.

Cell Type Figure 15 shows the distribution of the combinational and sequential cells for each cipher within the top-50% of the ranked cells. With only top-1% of the ranked cells, we observe that in the block cipher, all of the top-1% cells are combinational cells. This indicates that these combinational cells are the cause of the highest peaks in the power variances of AES and GIFT-COFB. We see the number of the sequential cells remains constant until it reaches top-25% for AES, and top-35% for GIFT-COFB. The gap between the number of combinational

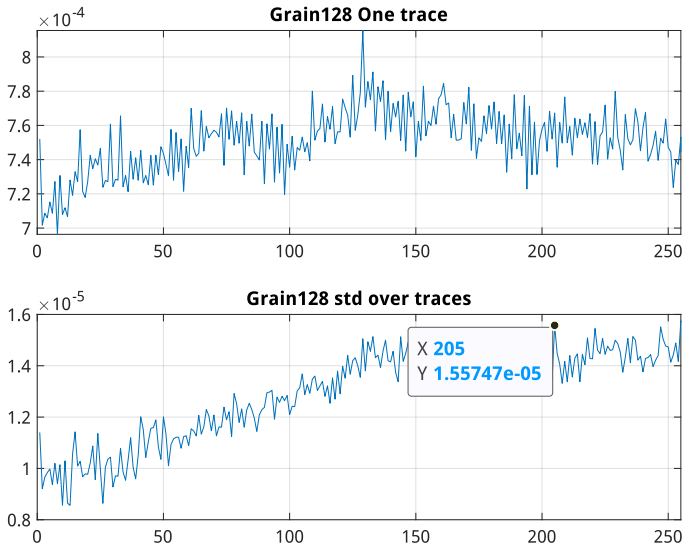

Fig. 13. (Top) Grain128-AEAD Power trace on one test vector (Bottom) Standard Deviation of Power over 256 power traces of Grain128-AEAD implementation

and sequential cells becomes larger with more top-ranked cells being presented, indicating that the majority of the top-50% of the ranked cells are dominated by the combinational cells in the block cipher. This further validates that most of the power variances in the block cipher are coming from the combinational cells.

On the other hand, we observe that the number of combinational and sequential cells in the stream cipher is relatively close to each other with top-1% of the ranked cells. But with more top-ranked cells being added, we see a drastic increase in the number of sequential cells. This indicates that the top-50% of the cells are dominated by sequential cells, validating that the most of the power variations in the stream cipher are caused by sequential cells.

In the sponge-based cipher, the gap between the combinational and sequential cells is the smallest among all ciphers. Therefore, the sponge-based cipher has both combinational and sequential cells contributing to the power standard deviation with the combinational cells dominating most of the power variances.

Figure 16 shows the impact on one cycle of power standard deviation after 1% to 30% of the cells are removed for all ciphers. We observe that top-5% of the ranked cells are causing the largest power variances for each cipher.

Area Table 2 shows the total area (in square micrometer) for each cipher using Cadence Genus. The area of AES is approximately three times larger than the

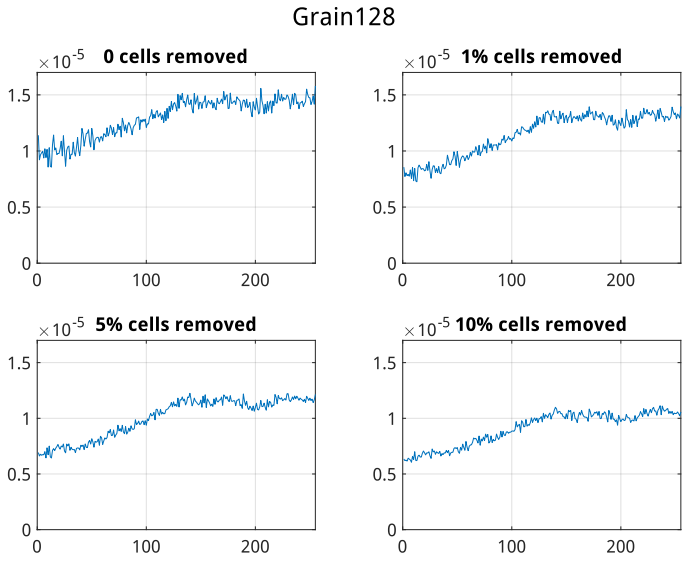

Fig. 14. Impact on Power Standard Deviation after removing 1%, 5% and 10% of the cells

Table 2. Synthesis Area for each Cipher

Cipher.	Total Area (μm^2).
AES	166k
GIFT-COFB	52k
Xoodyak	51k
Grain128-AEAD	14k

area of GIFT-COFB and Xoodyak. The area of Grain128-AEAD is almost twelve times smaller than that of AES.

To further analyze how much of the power variances have reduced with top-ranked cells removed, we plot the area under the curve of power standard deviation in both absolute and relative terms in Figure 17. We observe that after 35% of the ranked cells, the area under the curve in AES is close to zero in both figures. This means the top 35% of the cells in AES is contributing the majority of the side-channel leakage. The area reduction of GIFT-COFB has a similar slope as that of AES in the left figure. But the area under the curve of GIFT-COFB does not reach zero after 35% of the ranked cells, and it continues to decrease in both figures. This indicates that there are more cells in GIFT-COFB contributing to the power variances than that of AES in a relative term.

In Figure 17, left, we observe that Xoodyak and Grain128-AEAD have a more gentle area reduction slope compared to AES and GIFT-COFB. Xoodyak has

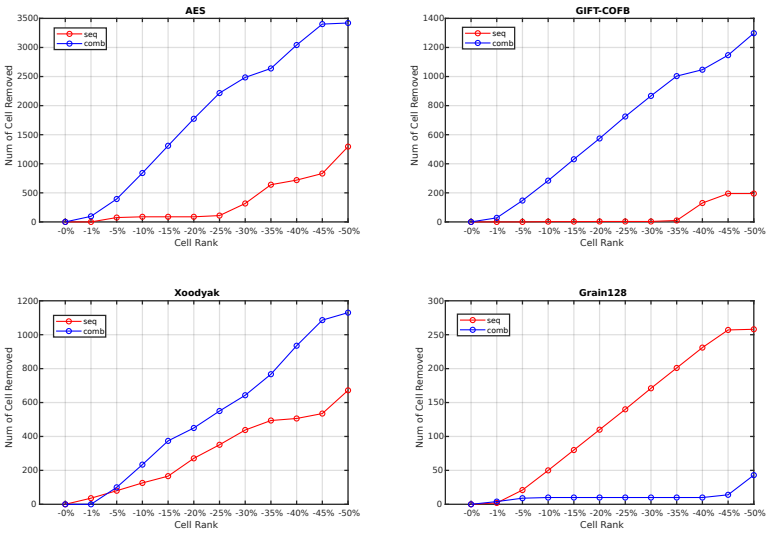

Fig. 15. Cell Type vs. Cell Rank by Percentage on AES, GIFT-COFB, Xoodyak, and Grain128-AEAD

more than 20% of the area left after 50% of the ranked cells in the right figure. This indicates that Xoodyak has the largest number of cells contributing to the power variances out of all ciphers in a relative term. With 5% of the ranked cells, AES drops almost 50% of the area, and the lightweight cryptography candidates drop almost 20% to 30% of the area. This validates that with our cell ranking strategy, top 5% of the cells are contributing to the highest power variances for each cipher.

Table 3. Percentage of Power Consumption for each Cipher

Cipher.	Register.	Logic.	Clock.
AES	28.71%	66.86%	4.43%
GIFT-COFB	24.00%	73.80%	2.20%
Xoodyak	35.00%	61.25%	3.75%
Grain128-AEAD	85.19%	3.54%	11.27%

Power Consumption Table 3 shows the relative power consumption of logic, register, and clock for each cipher. Power is mostly consumed by logic in AES, GIFT-COFB, and Xoodyak. Stream cipher has the highest power consumption in register and clock.

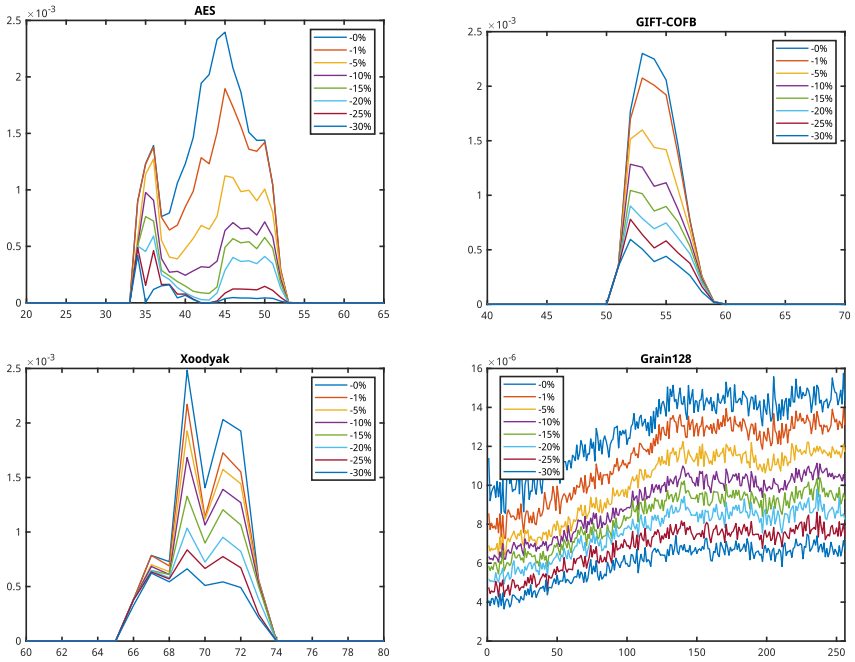

Fig. 16. Impact on One Cycle of Power Standard Deviation on AES, GIFT-COFB, Xoodyak, and Grain128-AEAD after removing up to 30% of the cells

5 Conclusions

We present a root cause analysis of power based side channel leakage in AES, GIFT-COFB, Xoodyak, and Grain128-AEAD. The novelty of this methodology is that we do not use a traditional power model. Instead, we choose our leakage model based on the power standard deviation of each cipher. We introduce a cell selection and ranking methodology to identify cells that cause the largest power variations. We validate the quality of the ranked cells by demonstrating the reduction in power standard deviations after top-ranked cells are removed. We further validate the quality of our methodology by performing CPA on AES with cells being removed according to ranking. We observe that in block cipher and sponge-based cipher, combinational cells are dominating the majority of the power variances. Sequential cells are the cause for most of the power variances in the stream cipher. We see that half of all power variation is contained within top-5% of the cells for all ciphers.

References

BBYS21. Ileana Buhan, Lejla Batina, Yuval Yarom, and Patrick Schaumont. Sok: Design tools for side-channel-aware implementations. *IACR Cryptol.*

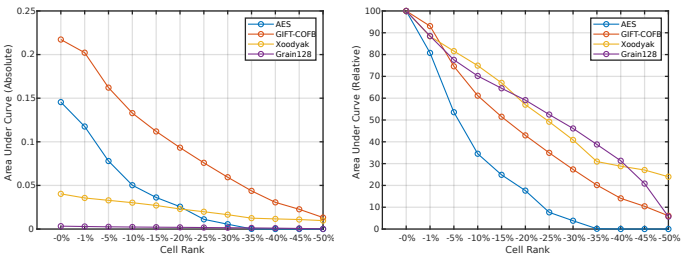

Fig. 17. Area Under the Curve of Power Standard Deviation vs. Cell Ranking by Percentage on AES, GIFT-COFB, Xoodyak, and Grain128-AEAD in Absolute terms (Left) and Relative terms (Right)

ePrint Arch., page 497, 2021.

BCI+20. Subhadeep Banik, Avik Chakraborti, Tetsu Iwata, Kazuhiko Minematsu, Mridul Nandi, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. Gift-cofb. *Cryptology ePrint Archive*, 2020.

DCBG+17. Thomas De Cnudde, Begül Bilgin, Benedikt Gierlichs, Ventzislav Nikov, Svetla Nikova, and Vincent Rijmen. Does coupling affect the security of masked implementations? In Sylvain Guilley, editor, *Constructive Side-Channel Analysis and Secure Design*, pages 1–18, Cham, 2017. Springer International Publishing.

DHP+20. Joan Daemen, Seth Hoffert, Michaël Peeters, G Van Assche, and R Van Keer. Xoodyak, a lightweight cryptographic scheme. 2020.

NRR06. Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *Information and Communications Security, 8th International Conference, ICICS 2006, Raleigh, NC, USA, December 4-7, 2006, Proceedings*, volume 4307 of *Lecture Notes in Computer Science*, pages 529–545. Springer, 2006.

PGA+22. Kostas Papagiannopoulos, Ognjen Glamocanin, Melissa Azouaoui, Dorian Ros, Francesco Regazzoni, and Mirjana Stojilovic. The side-channel metric cheat sheet, 2022.

SHSK19. Jonathan Sönnerup, Martin Hell, Mattias Sönnerup, and Ripudaman Khattar. Efficient hardware implementations of grain-128aead. In *International Conference on Cryptology in India*, pages 495–513. Springer, 2019.

WS17. Chao Wang and Patrick Schaumont. Security by compilation: an automated approach to comprehensive side-channel resistance. *ACM SIGLOG News*, 4(2):76–89, 2017.