Location, location, location: Revisiting modeling and exploitation for location-based side channel leakages

Christos Andrikos[1], Lejla Batina[2], Lukasz Chmielewski[2,4], Liran Lerman[5], Vasilios Mavroudis[6], Kostas Papagiannopoulos[2,4], Guilherme Perin[5], Giorgos Rassias[1] and Alberto Sonnino[6]

[1] National Techical University of Athens
christoscandrikos@cslab.ece.ntua.gr, grassias@cslab.ece.ntua.gr
[2] Radboud University
lejla@cs.ru.nl
[3] NXP Semiconductors Hamburg
kostaspap88@gmail.com
[4] Riscure BV
chmielewski@riscure.com, guilhermeperin7@gmail.com
[5] Thales Group
liran.lerman@be.thalesgroup.com
[6] University College London
v.mavroudis@cs.ucl.ac.uk, alberto.sonnino@ucl.ac.uk

Abstract. Near-field microprobes have the capability to isolate small regions of a chip surface and enable precise measurements with high spatial resolution. Being able to distinguish the activity of small regions has given rise to the location-based side-channel attacks, which exploit the spatial dependencies of cryptographic algorithms in order to recover the secret key. Given the fairly uncharted nature of such leakages, this work revisits the location side-channel to broaden our modeling and exploitation capabilities. Our contribution is threefold. First, we provide a simple spatial model that partially captures the effect of location-based leakages. We use the newly established model to simulate the leakage of different scenarios/countermeasures and follow an information-theoretic approach to evaluate the security level achieved in every case. Second, we perform the first successful location-based attack on the SRAM of a modern ARM Cortex-M4 chip, using standard techniques such as difference of means and multivariate template attacks. Third, we put forward neural networks as classifiers that exploit the location side-channel and showcase their effectiveness on ARM Cortex-M4, especially in the context of single-shot attacks and small memory regions. Template attacks and neural network classifiers are able to reach high spacial accuracy, distinguishing between 2 SRAM regions of 128 bytes each with 100% success rate and distinguishing even between 256 SRAM byte-regions with 32% success rate. Such improved exploitation capabilities revitalize the interest for location vulnerabilities on various implementations, ranging from RSA/ECC with large memory footprint, to lookup-table-based AES with smaller memory usage.

Keywords: Side-channel analysis · location leakage · microprobe · template attack · neural network · ARM Cortex-M

1 Introduction

Side-channel analysis (SCA) allows adversaries to recover sensitive information, by observing and analyzing the physical characteristics and emanations of a cryptographic implementation. Usually, physical observables such as the power consumption and electromagnetic (EM) emission of a device [24, 13] are closely related to the data that is being accessed, stored or processed. Such *data-based leakage* compromises the device's security and may allow the adversary to infer the implemented cipher's secret key.

Location-based leakage is a less common form of side-channel leakage when compared to data-based leakages, yet it arises in many practical scenarios. This form of leakage stems from the fact that chip components such as registers, memory regions, storage units, as well as their respective addressing mechanisms (control logic, buses) exhibit leakage when accessed and such leakage is identifiable and data-independent. Thus, the power or EM side-channel potentially conveys information about the *location* of the accessed component, i.e. it can reveal the particular register or memory address that has been accessed, regardless of the data stored in it. If there exists any dependence between the secret key and the location of the activated component, then a side-channel adversary can exploit it to his advantage and recover the key.

1.1 Previous Research & Terminology

The work of Sugawara et al. [48] demonstrates the presence of location-based leakage in an ASIC. In particular, they show that the power consumption of the chip's SRAM conveys information about the memory address that is being accessed. They refer to this effect as "geometric" leakage since it relates to the memory layout. Similarly, Andrikos et al. [2] performed preliminary analyses using the EM-based location leakage exhibited at the SRAM of an ARM Cortex-M4. The work of Heyszl et al. [18] manages to recover the secret scalar by exploiting the spatial dependencies of the double-and-add-always algorithm for elliptic curve cryptography. The experiments were carried out on a decapsulated FPGA, using near-field microprobes that identify the accessed register. Schlösser et al. [40] use the photonic side-channel in order to recover the exact SRAM location that is accessed during the activation of an AES Sbox lookup table. This location information can assist in key recovery, thus even cases of photonic emission analysis can be classified as location-based leakage. Moreover, countermeasures such as RSM [31] rely on rotating lookup tables to mask the data. Location-based leakage can identify which lookup table is currently under use and potentially weaken masking.

For the sake of clarity, we distinguish between "location leakage" and "localized leakage". Location leakage arises when knowing the location of a component (register, memory region, etc.) is assisting towards key recovery. On the contrary, localized leakage arises when the adversary is able to focus on the leakage of a specific (usually small) region of the chip. For example, recovering the memory address accessed during an Sbox lookup implies location leakage. Being able to measure the leakage right on top of a processor's register file implies that the adversary is capturing localized leakage. Note that capturing localized leakage can be useful for data-based attacks as well as for location-based attacks. The works of Unterstein et al. [51], Immler et al. [20] and Specht et al. [45, 44, 43] acquire localized leakage via a microprobe in order to improve the signal-to-noise ratio of their data-dependent leakage. The work of Heyszl et al. [18] uses the same technique in order to improve the signal-to-noise ratio of their location-dependent leakage. The current work follows experimental techniques similar to Heyszl et al. (localized EM) to showcase a potent location-based attack on ARM Cortex-M4 devices.

Again, for the sake of clarity we distinguish between "location leakage" and "address leakage" [21]. In our work, address leakage implies the leakage of addressing mechanisms, e.g. the leakage of the control logic of a storage unit. Such leakage can even be observed far from the storage unit itself, e.g. at memory buses or at the CPU. Location leakage implies the leakage caused by such address leakage *and* the leakage of the unit itself, which is often observed near it. We refer to the latter as "spatial leakage", i.e. location leakage encapsulates both address-related and spatial effects. For example accessing a table in memory requires indexing and memory addressing in the CPU (address leakage). In addition, accessing causes the memory itself to be activated (spatial leakage). The adversary is usually able to observe both types of leakage and it is often hard to distinguish between them.

1.2 Contribution & Organization

This work presents the following results in the field on location-based leakage by expanding our modeling and exploitation capabilities.

1. We provide a simple model that captures the effect of spatial leakages. The model is motivated by experimental data observed in the SRAM of an ARM Cortex-M4.
2. Using the newly established model, we simulate the different theoretical scenarios that enhance or diminish spatial leakage. We investigate the security of every scenario using the perceived information (PI) metric.
3. We perform the first practical location-based attack on the SRAM of a modern ARM Cortex-M4, using difference-of-means, multivariate template attacks and neural networks.
4. We showcase attacks where it is possible to distinguish consecutive SRAM regions of 128 bytes each, with 100% success rate and to distinguish between 256 consecutive SRAM bytes with 32% success rate. We conclude that EM location-based leakages are potent enough to compromise the security of AES implementations that use SRAM lookup-tables.

Notation. Capital letters denote random variables and small case letters denote instances of random variables or constant values. Bold font denotes vectors. For instance, side-channel leakage variables are denoted by L and their instances by l; and likewise leakage vectors are denoted by $\mathbf{L}$ and their instances by $\mathbf{l}$. The notation $Unif(\{a, b\})$, $Bern(p)$, $Binomial(n, p)$ and finally $Norm(\mu, \sigma^2)$ denotes random variables with uniform, Bernoulli, binomial and normal probability distributions respectively. Parameter p denotes the probability of Bernoulli/binomial trials and μ, σ^2 denote the mean and variance of the normal distribution. The set $\{a, b\}$ denotes that the discrete uniform distribution can receive value a or b equiprobably. The notation $E[\cdot]$, $Var[\cdot]$ and $H[\cdot]$ describes the expected value, variance and entropy of a random variable. Finally, the notation $H_{p,q}[\cdot]$ shows the cross entropy of a random variable, between probability distributions p and q.

Organization. Section 2 describes the microprobe-based experimental setup on ARM Cortex-M4, shows a simple location analysis using difference-of-means, and motivates experimentally the spatial part of location leakage. Section 3 puts forward the spatial leakage model, describes several theoretical scenarios, and performs an evaluation using the perceived information metric. Section 4 demonstrates real-world template attacks on ARM Cortex-M4 for various cases and Section 5 demonstrates the attacks using neural networks on the same device. We conclude and discuss future directions in Section 6.

2 Experimental Setup & T-Test Analysis

This section describes a high-precision EM-based setup that is able to detect location leakage on the surface of an ARM Cortex-M4 (Sections 2.1, 2.2). Using the setup, we obtain intuition about the location leakage that is caused by switching circuitry and is observable via EM emissions on the die surface (Section 2.3). Throughout the text, we concentrate on the following adversarial scenario. The device has implemented a key-dependent cipher operation that uses a lookup-table and the adversary aims to infer which part of the table is active, i.e. uncover the location information leading to key recovery.

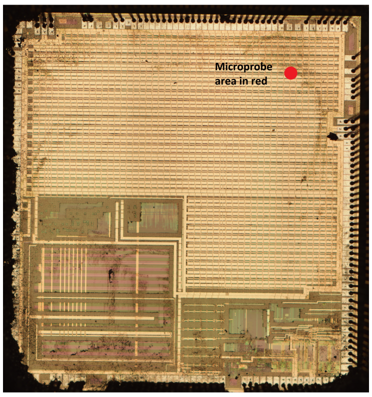


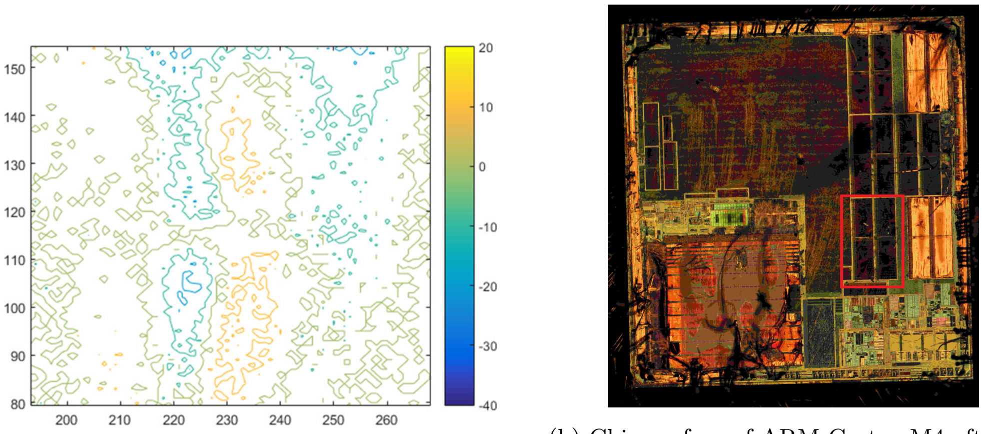
Figure 1: The chip surface of the device-under-test (ARM Cortex-M4) after removal of the plastic layer. The approximate area of the ICR HH 100-27 Langer microprobe is shown by the red circle (0.03 mm^2).

2.1 Experimental Setup

The main goal of our experimental evaluation is to examine whether it is possible to detect the access to different SRAM regions in a modern ARM-based device. Rephrasing, we examine the device's susceptibility to location-based attacks during e.g. key-dependent memory lookups, similarly to AES LUT. Our measurement setup consists of a decapsulated Riscure Piñata device[1], on a modified board, fabricated with 90 nm technology. The decapsulated chip surface (roughly 6 $mm^2 \approx 2.4\ mm \times\ 2.4\ mm$) is scanned using an ICR HH 100-27 Langer microprobe[2] with diameter of 100 μm (approximately 0.03 mm^2). The scan is performed on a rectangular grid of dimension 300, using the Inspector tooling[3] and resulting in 300 × 300 measurement spots. The near-field probe is moved over the chip surface with the assistance of an XYZ-table with positioning accuracy of 50 μm. At every position of the scan grid, a single measurement is performed, using sampling rate of 1 Gsample/sec and resulting in 170k samples. Due to the complex and non-homogeneous nature of a modern chip, several types of EM emissions are present on the surface, most of which are unrelated to the SRAM location. In this particular case study, the signals of interest were observed in amplitudes of roughly 70 mV, so we set the oscilloscope voltage range accordingly. In addition, several device peripherals (such as USB communication) have been disabled in order to reduce interference. The decapsulated surface where the scan is performed is visible in Figure 1 and the approximate microprobe area is also overlaid on the figure (in red) for comparison.

To effectively cause location-dependent leakage, we perform sequential accesses to a continuous region of 16 KBytes in the SRAM by loading data from all memory positions. The data at all accessed memory positions have been fixed to value zero prior to the experiment in order to remove any data-based leakage. The word size of this ARM architecture is 32 bits, i.e. we accessed 4096 words in memory. We opted to access the SRAM using ARM assembly instead of a high-level language in order to avoid compiler-induced optimizations that could alter the side-channel behavior

[1] https://tinyurl.com/y9tmnklr
[2] https://tinyurl.com/mcd3ntp
[3] https://tinyurl.com/jlgfx95



(a) Distinguishing two 8 KByte regions of the SRAM with difference-of-means. Yellow region indicates stronger leakage from class 1 while blue region from class 2.

(b) Chip surface of ARM Cortex-M4 after removal of the top metal layer. The red rectangular region corresponds to the difference-of-means plot of Figure 2a, i.e. it shows the location where the highest differences were observed.

Figure 2: Spatial properties of chip leakage.

2.2 Difference-of-Means T-Test

The initial scan measurements were analyzed using a simple difference-of-means test. To demonstrate the presence of location-based leakage, we partitioned every trace (170k samples) into two classes. The first class contains SRAM accesses from the beginning of the memory until word no. 2047 and the second class contains SRAM accesses from word 2048 until word 4096. Each class corresponds to 8 KBytes of SRAM. For every grid position (x, y), we averaged the leakages samples of class 1 and class 2 producing $\bar{l}_{class1} = \frac{1}{85k} \sum_{j=1}^{85k} l_{x,y}^{j}$ and $\bar{l}_{class2} = \frac{1}{85k} \sum_{j=85k}^{170k} l_{x,y}^{j}$ respectively. Continuing, we computed the difference of means $\bar{l}_{class1} - \bar{l}_{class2}$ and we performed a Welch t-test with significance level of 0.1% in order to determine if location-based leakage is present. The results are visible in Figure 2a, which is focusing on a specific part of the chip surface that exhibits high difference.

2.3 Motivating the Location Leakage Model

In Figure 2a we can observe that the spatial part of location leakage is indeed present in the ARM Cortex-M4 and it can even be detected through simple visual inspection if memory regions are large enough (8 KBytes). Repeating the same difference-of-means test for SRAM regions of 4 KBytes yields similar results, i.e. the regions remain visually distinct. In both cases, we observe that these location dependencies demonstrate strong *spatial characteristics*. That is, in Figure 2a we see two regions at close proximity (yellow and blue) where the yellow region shows positive difference between class 1 and 2, while the blue region shows negative difference between class 1 and 2. To investigate this proximity, we performed additional chemical etching on the chip surface in order to remove the top metal layer (Figure 2b).

The different regions (yellow, blue) shown in Figure 2a are observed directly above the chip area enclosed by the red rectangle of Figure 2b. Interestingly, after the removal of the top metal layer, we see that the red rectangular region contains large continuous chip components, possibly indicating that SRAM circuitry is present at this location. This hypothesis is corroborated by the following fact: when we perform difference-of-means test for 4 KByte regions, the yellow and blue regions shrink, indicating that the leakage area is *proportional* to the memory size that is being activated.

The approximate surface area of an SRAM component can be estimated as $a = \frac{m \cdot a_{bit}}{e}$, where m is the number of bits in the memory region, a_{bit} is the area of a single-bit memory cell and e is the array layout efficiency (usually around 70%) [54]. The value of a_{bit} ranges from $600\lambda^2$ to $1000\lambda^2$, where λ is equal to half the feature size, i.e. for the current device-under-test $\lambda = 0.5 * 90\ nm$, thus the area of a 32-bit word is between 55 and 92 μm^2. Likewise, an 8 KByte region of the ARM Cortex-M4 amounts to an area of approximately 0.12 until 0.19 mm^2, depending on the fabrication process. Notably, this area estimation is quite close to the area of the yellow or the blue region of Figure 2a (approximately half of the red rectangle). Similar spatial characteristics have been observed by Heyszl et al. [18] in the context of FPGA registers.

Thus, experimental evidence that suggest that A) proximity exists between leaky regions and B) the area of leaky regions is approximately proportional to the memory size that we activate. Section 3 builds up on these observations and develops a simple model that describes *spatial leakage*, yet we first need to provide the following disclaimer.

Word of caution. The activation of a memory region can indeed be inferred by observing spatial leakage, which according to experimental data is quite rich in location information. Still, this does not imply that spatial leakage is the *sole* source of location leakage. It is possible that location information is also revealed through address leakage on the CPU and the memory control logic or buses when they process SRAM addresses, or even by other effects such as imperfect routing [52]. Thus, modeling spatial leakage captures *part* of the available information and can be considered as the first step towards full modeling of location leakage.

3 A Spatial Model For Location Leakage

Unlike the well-established power and EM data leakage models [12, 46], high-resolution EM-based location leakage remains less explored. The main reason is the semi-invasive nature of location attacks (often requiring chemical decapsulation), the time-consuming chip surface scanning and the lengthy measurement procedures involved. Still, we maintain that such attacks are increasingly relevant due to the fairly average cost (approx. 15k euros), along with the widespread protection against data leakages [39, 33], which encourages attackers towards different exploitation strategies.

Hence, this section puts forward a theoretical model that describes the spatial part of location leakage on a chip surface. The model can be viewed as an extension of the standard data-based model to the spatial domain, encapsulating the complexity of surface-scanning experiments. The proposed simulation of Section 3.1, in conjunction with the analysis of Section 3.2 can significantly enhance the design and evaluation cycle of SCA-resistant devices. Our approach allows the countermeasure designer to gauge the amount of experimental work an adversary would need to breach the device using spatial leakage. Thus, the designer can fine-tune protection mechanisms, provide customized security and avoid lengthy design-evaluation cycles by capturing certain security hazards at an early stage. The time-consuming leakage certification on the physical device can be carried out at a later stage, once obvious defects have been fixed. Naturally, all simulation-driven models (including this work) have inherent limitations, i.e. they are incapable to describe all the underlying physical phenomena, as we shall see in Section 4. Still, avoiding core issues early on, can free up valuable time that evaluators can invest towards device-specific effects such as coupling [10] and leakage combination [35].

3.1 Model Definition and Assumptions

Experimental Parameters. We define a side-channel experiment ϵ as any valid instance of the random variable set $\mathcal{E} = \{S, O, G, \mathbf{A}, \mathbf{P}\}$. The experimental parameters are shown in Table 1. We designate the experiment's goal to be the acquisition of spatial leakage $\mathbf{L}$, i.e. obtain $(\mathbf{L}|\mathcal{E} = \epsilon)$ or $(\mathbf{L}|\epsilon)$ for short. Much like in Section 2, the experiment consists of a probe scan over the chip surface in order to distinguish between different components (or regions) and ultimately between different memory addresses, registers, etc. The parameter S denotes the *area of the chip surface* on which we perform measurements, e.g. s can be the whole chip die (6 mm^2) or any smaller surface. Parameter O denotes the *area of the measuring probe* that we use in our experiments, e.g. the area o of the ICR HH 100-27 microprobe is roughly 0.03 mm^2. Continuing, parameter G denotes the *measurement grid dimensions*, i.e. it specifies the resolution of a uniform rectangular array of antennas [53]. In Section 2 we opted for $g = 300$. Continuing, the vector parameters $\mathbf{A}, \mathbf{P}$ describe the n_c surface components that emit EM-based spatial leakage. The parameter $\mathbf{A} = [A_1, A_2, \ldots, A_{n_c}]$ describes the surface *area occupied by each component*, e.g. in Section 2.3 we estimated the area of an 32-bit word component to be at most 92 μm^2. The parameter $\mathbf{P} = [\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_{n_c}]$ describes the *position of every component* on the chip surface, i.e. $\mathbf{P}_i$ is a 2-dimensional vector. For simplicity, we assume the geometry of the surface, probe and components to be square, yet we note that the model can be extended to different geometrical shapes in a straightforward manner. Moreover, we assume that the measuring probe can capture only emissions that are directly beneath it, i.e. it functions like an identity spatial filter with area o.

Table 1: Parameters of simulated experiment

Parameter	Description	Unit
S	chip surface area	u^2
O	probe area	u^2
G	scan grid dimension	<no unit>
A	component areas	vector with 1D entries of u^2
P	component positions	vector with 2D entries of u

Control Parameter. Every device can use program code to activate different components of the chip surface, e.g. by accessing different SRAM words through load/store instructions. To describe this, we use an additional control parameter $\mathbf{C}$ that denotes which components (indexed $1, \ldots, n_c$) are accessed during a particular experiment ϵ. Analytically, $\mathbf{C} = [C_1, C_2, \ldots, C_{n_c}]$, where $C_i = 1$ if component i is active during the experiment and $C_i = 0$ if it is inactive; for instance the vector $\mathbf{c} = [0, 1, 0]$ implies that the surface has 3 components ($n_c = 3$) and only component no. 2 is currently active. Note also that in our model only one out of n_c components can be active at a given point in time, since we assume that the ordinary microcontrollers do not support concurrent memory access.[4]. Thus the parameter $\mathbf{c}$ uses the one-hot encoding and we define $\mathbf{v}^i$ as an n_c-dimensional vector where all entries are zero except for the i^{th} entry. For instance, if $n_c = 3$, then $\mathbf{v}^3$ is equal to $[0, 0, 1]$ and it describes the program state where only component no. 3 is active. In general, we use the notation $(\mathbf{L}|\mathcal{E} = \epsilon, \mathbf{C} = \mathbf{v}^i)$ or equivalently $(\mathbf{L}|\epsilon, \mathbf{v}^i)$ to describe a side-channel experiment ϵ that captures the leakage when the i^{th} component is active. For an attack to be successful, we need to distinguish between two (or more) different components using this spatial leakage. Formally, we need to be able to distinguish between $(\mathbf{L}|\epsilon, \mathbf{v}^i)$ and $(\mathbf{L}|\epsilon, \mathbf{v}^j)$, for $i \neq j$.

Representative Example. To elucidate the model, Figure 3 presents an experiment ϵ using parameters $\{s, o, g, \mathbf{a}, \mathbf{p}\}$ equal to $\{25, 3, 2, [0.8, 3], [[0.6, 1.5], [1.6, 4.1]]\}$, where all position parameters are in arbitrary units u and all area parameters are in square units

[4] Parallel word processing can be easily included.

u^2. The experiment targets two components ($n_c = 2$) and their position is $[0.6\ u, 1.5\ u]$ and $[1.6\ u, 4.1\ u]$ respectively. The surface area s, probe area o, and component areas a_1 and a_2 are respectively 25 u^2, 3 u^2, 0.8 u^2 and 3 u^2. The dimension g of the measurement grid is 2, resulting in a 2×2 scan and we capture a single measurement (trace) in every grid spot. We use the program code (control parameter) to activate components 1 and 2, generating $(\mathbf{L}|\epsilon, [1, 0])$ and $(\mathbf{L}|\epsilon, [0, 1])$ respectively. Note that in general $(\mathbf{L}|\epsilon, \mathbf{v}^i)$ results in leakage with g^2 dimensions, e.g. $(\mathbf{L}|\epsilon, [1, 0])$ is a 4-dimensional vector. We refer to the leakage measured at any specified position $[x, y]$ as $(L_{[x,y]}|\epsilon, \mathbf{v}^i)$ or simply $L_{[x,y]}$.

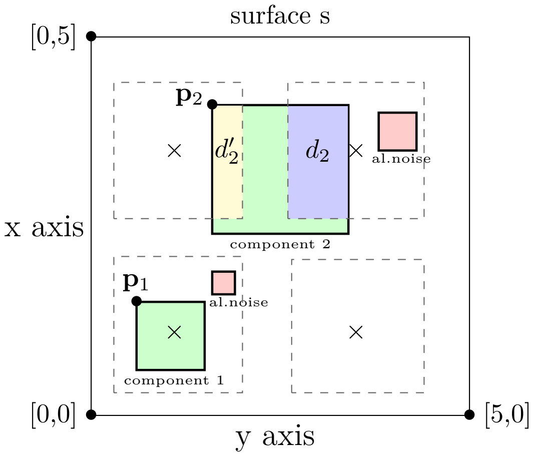


Figure 3: Sample experiment ϵ. The × spots show the measurement points of the 2×2 scan grid. Dashed black-line rectangles enclosing these spots denote the measuring probe area o. Vectors $\mathbf{p}_1, \mathbf{p}_2$ show the position of two components ($n_c = 2$), whose areas (a_1, a_2) are enclosed by the solid black-line rectangles. The blue area d_2 shows the area of component 2 captured by the top-right measurement point and the yellow area d_2' shows the area of component 2 captured by the top-left measurement point.

Independent Noise. In accordance with standard data-based leakage models, we assume that for given parameters $\epsilon, \mathbf{v}^i$, the leakage $L_{[x,y]}$ at any grid position $[x, y]$ consists of a deterministic part $l^{det}_{[x,y]}$, an algorithmic noise part N^{algo} and an electrical noise part N^{el}, thus: $L_{[x,y]} = l^{det}_{[x,y]} + N^{algo} + N^{el}$

Deterministic Leakage. We assume that the deterministic part of the leakage $l^{det}_{[x,y]}$ at position $[x, y]$ is caused by the activation (switching behavior) of any component that is captured by the probe at this grid position. Based on the experimental observations of Section 2.3, we assume the deterministic leakage to be proportional to the area of the active component located underneath the probe surface, thus:

$$l^{det}_{[x,y]}|\mathbf{v}^i = \begin{cases} 0, & \text{if comp. } i \text{ is not captured at [x,y]} \\ d_i,\ 0 < d_i < a_i, & \text{if comp. } i \text{ is partially} \\ & \quad \text{captured at [x,y]} \\ a_i, & \text{if comp. } i \text{ is fully captured at [x,y]} \end{cases}$$

For example, Figure 3 shows that component 1 is fully captured by the probe on the bottom-left grid spot, thus $(l^{det}_{[down,left]}|\mathbf{v}^1) = a_1$. Since no other measurement position can capture component 1, it holds that $(l^{det}|\mathbf{v}^1) = 0$ for the other three grid positions. On the contrary, component 2 is partially captured in two grid positions. Thus, it holds that $(l^{det}_{[up,right]}|\mathbf{v}^2) = d_2$ (blue area), $(l^{det}_{[up,left]}|\mathbf{v}^2) = d_2'$ (yellow area) and zero elsewhere.

Electrical and Algorithmic Noise. We employ the common assumption that the electrical noise N^{el} follows a normal distribution with zero mean and variance σ_{el}^2, i.e. $N^{el} \sim Norm(0, \sigma_{el}^2)$. The variance σ_{el}^2 is related to the specific device-under-test and measurement apparatus that we use.

The algorithmic noise in our model is caused by components that, like the targeted components, leak underneath the probe on measurement spot of the scan grid. However, unlike our targeted components, they exhibit uniformly random switching activity (equiprobable 'on' and 'off' states) that is independent of the control parameter $\mathbf{c}$. If n_a such components, with area parameter $\mathbf{b} = [b_1, b_2, \ldots, b_{n_a}]$ are located under the probe, then we assume again their leakage to be proportional to the respective captured area. The leakage of these independent, noise-generating components is denoted by N_i^{algo}, $i = 1, \ldots, n_a$. Thus, N^{algo} constitutes of the following sum.

$$N^{algo} = \sum_{i=1}^{n_a} N_i^{algo}, \text{where } N_i^{algo} \sim Unif(\{0, b_i\})$$

The algorithmic noise is highly dependent on the device-under-test, i.e. we could potentially encounter cases where there is little or no random switching activity around the critical (targeted) components, or we may face tightly packed implementations that induce such noise in large quantities. For example, in Figure 3 the top-left and bottom-right spots have no algorithmic noise, while the top-right and bottom-left spots contain randomly switching components (red rectangles) that induce noise. Note that the larger the probe area o, the more likely we are to capture leakage from such components. Appendix A elaborates on the form of algorithmic noise on tightly-packed surfaces.

3.2 Information-Theoretic Analysis

The proposed spatial leakage model of Section 3.1 is able to simulate the EM emission over a chip surface and provide us with side-channel observables. Due to the complexity of surface-scanning experiments, the model needs to take into account multiple parameters in ϵ (component area, grid size, noise level, etc.), all of which can directly impact our ability to distinguish between different regions.

In order to demonstrate and gauge the impact of the experimental parameters on the side-channel security level, this section introduces an information-theoretic apporach to analyze the following simple location-leakage scenario. Using the model of Section 3.1, we simulate the spatial leakage emitted by the ARM Cortex-M4 SRAM, while accessing a lookup-table (LUT) of 256 bytes. This LUT computation, emulates the spatial leakage of an AES LUT, while excluding any data-based leakages. The processor uses a 32-bit architecture, thus we represent the 256-byte lookup table with 64 words (4 bytes each) stored consecutively in SRAM. The LUT memory region is placed randomly[5] on a chip surface with $s = 0.6\ mm^2$. Then our model generates leakage stemming from 64 chip components ($n_c = 64$), where each one occupies surface area pertaining to 4 SRAM bytes. Using the simulated traceset, we perform template attacks [6] after PCA-based dimensionality reduction [3], in order to distinguish between different LUT regions (consisting of one or more words). Being able to infer which LUT/SRAM region was accessed can substantially reduce the number of AES key candidates. For instance, the adversary may template separately the leakage of all 64 words (high granularity) in order to recover the exact activated word and reduce the possible AES key candidates from 256 to 4. Alternatively, he can partition the LUT to two regions (words 0 until 31 and words 32 until 63), profile both regions (low granularity), in order to recover the activated 128-byte region and reduce the AES key candidates from 256 to 128.

[5] Unless specified otherwise, we place every word directly next to each other, starting from a random position in the surface.

Formally, at a certain point in time, the microcontroller is able to access only one out of 64 components (high granularity), thus the control variable $\mathbf{c} \in \mathcal{V} = \{\mathbf{v}^1, \mathbf{v}^2, \dots, \mathbf{v}^{64}\}$ and the adversary can observe the leakage of word-sized regions $(\mathbf{L}|\mathbf{C} = \mathbf{v}^i)$, for $i = 1, 2, \dots, 64$. Alternatively (low granularity), he can focus on $|\mathcal{R}|$ memory regions and partition the set $\mathcal{V}$ to sets $\mathcal{V}^1, \mathcal{V}^2, \dots, \mathcal{V}^{|\mathcal{R}|}$, where usually $\mathcal{V}^r \subset \mathcal{V}$ and $\mathcal{V}^i \cap \mathcal{V}^j = \emptyset$, for $i \neq j$. We define random variable $R \in \mathcal{R} = \{1, 2, \dots, k\}$ to denote the activated region and we represent the leakage of region r as $(\mathbf{L}|R = r) = (\mathbf{L}|\mathbf{c} \in \mathcal{V}^r)$. For example, in the high granularity scenario, the adversary observes and profiles $(\mathbf{L}|\mathbf{v}^1), (\mathbf{L}|\mathbf{v}^2), \dots, (\mathbf{L}|\mathbf{v}^{64})$, while in the low granularity scenario he profiles two regions ($\mathcal{R} = \{1, 2\}$) with $\mathcal{V}^1 = \{\mathbf{v}^1, \mathbf{v}^2, \dots, \mathbf{v}^{32}\}$ and $\mathcal{V}^2 = \{\mathbf{v}^{33}, \mathbf{v}^{34}, \dots, \mathbf{v}^{64}\}$. Thus he can obtain $(\mathbf{L}|R = 1) = (\mathbf{L}|\mathbf{c} \in \mathcal{V}^1)$ and $(\mathbf{L}|R = 2) = (\mathbf{L}|\mathbf{c} \in \mathcal{V}^2)$.

Having completed the profiling of regions for a certain experiment ϵ, we quantify the leakage, using the perceived information metric (PI) [38] as follows.

$$PI(\mathbf{L}; R) = H[R] - H_{true,model}[\mathbf{L}|R] = H[R] + \sum_{r \in \mathcal{R}} Pr[r] \cdot \int_{\mathbf{l} \in \mathcal{L}^{g^2}} Pr_{true}[\mathbf{l}|r] \cdot log_2 Pr_{model}[r|\mathbf{l}] \; d\mathbf{l}$$

$$Pr_{model}[r|\mathbf{l}] = \frac{Pr_{model}[\mathbf{l}|r]}{\sum_{r^* \in \mathcal{R}} Pr_{model}[\mathbf{l}|r^*]} Pr_{true}[\mathbf{l}|r] = \frac{1}{n_{test}}, \; n_{test} \text{ size of test set}$$

PI can quantify the amount of information that leakage $\mathbf{L}$ conveys about the activated region R, taking into account the divergence between the real and estimated distributions. Computing PI requires the distribution $Pr_{model}[\mathbf{l}|r]$, i.e. the template that is estimated from the training dataset. In addition, it requires the true leakage distribution $Pr_{true}[\mathbf{l}|r]$, which is unknown and can only be sampled directly from the test dataset. We opt for this metric since it indicates when degraded (under-trained) leakage models are present, due to our choice of experimental parameters. Negative PI values indicate that the trained model is incapable of distinguishing regions, while a positive value indicates a sound model that can lead to classification.

Using the proposed leakage simulation and the PI metric we evaluate several scenarios for the LUT case. Sections 3.2.1 until 3.2.5 showcase how different experimental parameters hinder or enhance leakage, offering several design options. To apply the theoretical model in an evaluation context we can simply set our current device SNR to the PI graphs.

3.2.1 Area and number of regions

The first simulation scenario examines the core attack question: using a certain experimental setup with parameters $\epsilon = \{s, o, g, \mathbf{a}, \mathbf{p}\}$, what is the smallest region size that I can distinguish reliably? Rephrasing, we assess how much location information can be extracted from the observed leakage by plotting the $PI(\mathbf{L}; R)$ metric against the electrical noise variance σ^2_{el} for certain ϵ and $\mathbf{c}$ parameters. We simulate an adversary that distinguishes regions of a 256-byte LUT using the following three LUT partitions of increasing granularity. First, he partitions the 256-byte LUT to 2 regions of 128 bytes each (depicted by the solid line in Figure 4). Second, he partitions the LUT to 8 regions of 32 bytes (dashed line) and third to 16 regions of 16 bytes (dotted line). For every partition we profile the regions' leakage $(\mathbf{L}|R = r) = (\mathbf{L}|\mathbf{c} \in \mathcal{V}^r)$ for $r = 1, 2, \dots, |\mathcal{R}|$, where $|\mathcal{R}| = 2$ or 8 or 16 and subsequently tries to distinguish. Note that surface $s = 6 \; mm^2$, probe size $o = 0.03 \; mm^2$ (ICR HH 100-27), feature size 90 nm and $g = 100$, i.e. the scan resolution is 100×100. The component area $a = 92 \; \mu m^2$ for all SRAM words and the words are placed adjacent to each other, starting from a random surface position; we denote this as $\mathbf{p} = random$. Along with parameters ϵ and $\mathbf{c}$, we need to include the measurement complexity in our simulation. Thus, we specify the amount of traces measured at every grid spot, resulting in an acquisition of $g^2 \cdot \#traces$. As expected, the experiments with higher region granularity

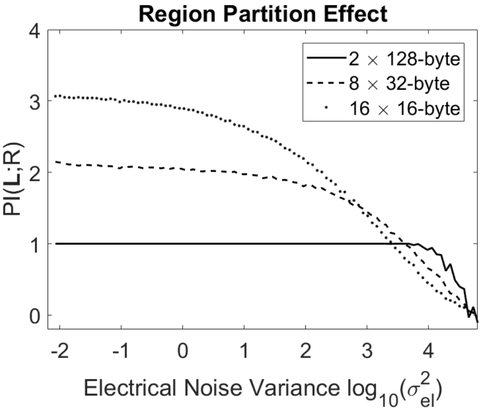


Figure 4: Region partition of 256-byte LUT to 2, 8, 16 regions. Parameters $\epsilon = \{6\ mm^2, 0.03\ mm^2, 100, 92\ \mu m^2, \text{random}\}$, capturing 10 traces /spot 100k traces in total.

yield more location information, as shown by the vertical gaps of the PI metric in Figure 4. Still, we also observe that smaller regions are harder to distinguish, even for low noise levels. Partitioning to 8 or 16 regions could optimally yield 3 or 4 bits of information respectively, yet the dashed and dotted curves remain well below this limit. Thus, we note that the adversary may need to improve his experiment ϵ by measuring more traces, using smaller probes or increasing the grid dimension in order to extract the maximum information.

3.2.2 Measurement grid dimension

Any side-channel experiment involving surface scanning can be particularly time-consuming. Moving the microprobe between adjacent positions takes approximately 2 seconds, thus the 300 × 300 surface scan carried out in Section 2 takes almost 2 days to conduct. Using the spatial leakage simulation, we can specify the grid dimension g and find the minimum scan resolution required to distinguish between certain SRAM regions. Figure 5 demonstrates the information captured when conducting scans with resolutions 100 × 100, 40 × 40 and 20 × 20, taking approximately 6 hours, 1 hour and 15 minutes respectively. Across the three simulations we maintain constant data complexity of 100k traces, distributed to grid spots accordingly (10, 62 and 250 traces per spot). Figure 5 shows information loss (vertical gap) as the grid dimension is decreasing, i.e. when trying to distinguish 4 regions only the 6 hour-experiment with 100 × 100 grid is able reach maximum information (2 bits). Notably, we see that for larger noise levels, small grid sizes with many traces per spot (dense measurements) are able to outperform larger grid sizes with less traces per spot (spread measurements).

3.2.3 Feature size

A common issue encountered in the side-channel literature is the scaling of attacks and countermeasures as devices become more complicated and feature size decreases [32, 22, 30]. This section uses our simple leakage model to describe the effect of feature size on SCA. We simulate the location leakage of SRAM cells fabricated with 180 nm, 120 nm and 90 nm technologies, resulting in bit cell areas of approximately 8 μm^2, 3.5 μm^2 and 2 μm^2. The results are visible in Figure 6. Naturally, smaller technology sizes can potentially limit the amount of available information, as they decrease the region's area and force the adversary towards more expensive tooling.

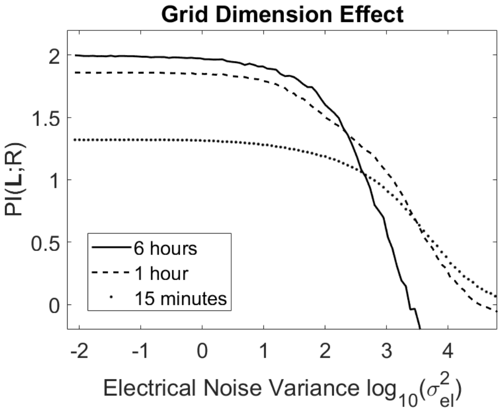


Figure 5: Grid dimension $g = 100, 40$ and 20. Parameters $\epsilon = \{6\ mm^2, 0.03\ mm^2, g, 92\ \mu m^2, \text{random}\}$, distinguishing 4 regions of 64 bytes each, 100k traces in total.

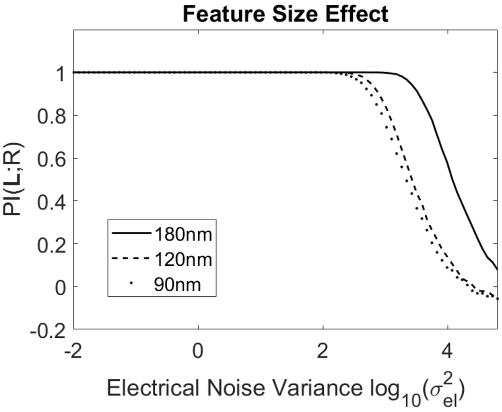


Figure 6: Feature size of 180, 120, 90 nm, word area $a = 368$, 163, 92 μm^2. Parameters $\epsilon = \{6\ mm^2, 0.03\ mm^2, 40, a, \text{random}\}$, for 2 regions of 128 bytes each, 250 measurements/spot, 400k traces in total.

3.2.4 Algorithmic noise

This section simulates the countermeasure of spatial algorithmic noise, when implemented on the ARM device. Analytically, we examine the case where the designer is able to place word-sized noise-generating components on the chip surface in order to “blur” the location leakage of a targeted region and hinder recovery. The simulation (Figure 7) uses the analysis of Appendix A to approximate the algorithmic noise when the probe captures the leakage of 11 SRAM words, one of which is the target word (and reveals the critical region information) and the ten remaining words are randomly activated at the same time. Observing Figure 7, we see the algorithmic-noise PI curve (dashed line) shifting to the left of the PI curve without algorithmic noise (solid line). Thus, much like data-based algorithmic noise [47], we see that randomly activating words functions indeed as an SCA countermeasure.

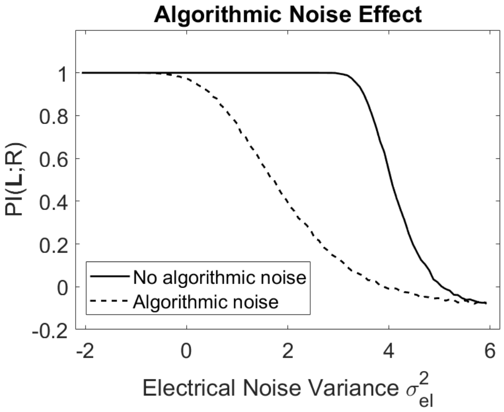


Figure 7: Deploying 10 noise-generating words. Parameters $\epsilon = \{6\ mm^2, 0.03\ mm^2, 40, 92\ \mu m^2, \text{random}\}$, for 2 regions of 128 bytes each, 250 measurements/spot, 400k traces in total.

3.2.5 Region proximity and interleaving

Last, we simulate the countermeasure of region proximity and region interleaving on the ARM device, which was considered by He et. al [16] and Heyszl [17]. Analytically, we assume that the designer controls the place-and-route process and can place two memory regions on the chip surface using the following three configurations.

1. Distant placement: the distance between the two regions is roughly 1 mm.
2. Close placement: adjacent placement of the two regions.
3. Interleaved placement: the words of the two regions are interleaved together in a checkered fashion, i.e. word 0,2,4,... of SRAM belonds to 1st region and word no. 1,3,5,... belongs to 2nd region.

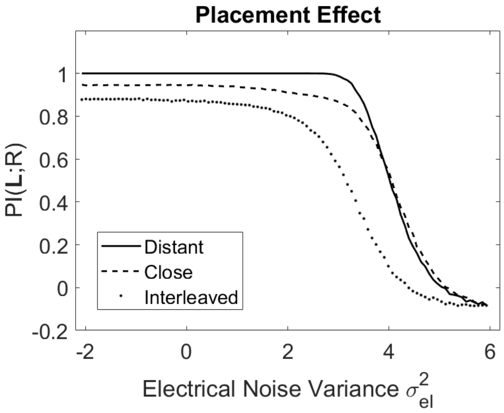


Figure 8: Distant, close, interleaved placement. Parameters $\epsilon = \{6\ mm^2, 0.03\ mm^2, 20, 92\ \mu m^2, \text{random}\}$, distinguishing 2 regions of 128 bytes each and using 250 measurements/spot, 100k traces in total.

Figure 8 demonstrates the effect of different placement choices, confirming the basic intuition that higher proximity is essentially a countermeasure against location-based leakage. The vertical gap in PI between distant, close and interleaved placement shows that as components get closer, the attainable information decreases, forcing the adversary to increase the grid size or use a smaller probe.

4 Exploitation Using Template Attacks

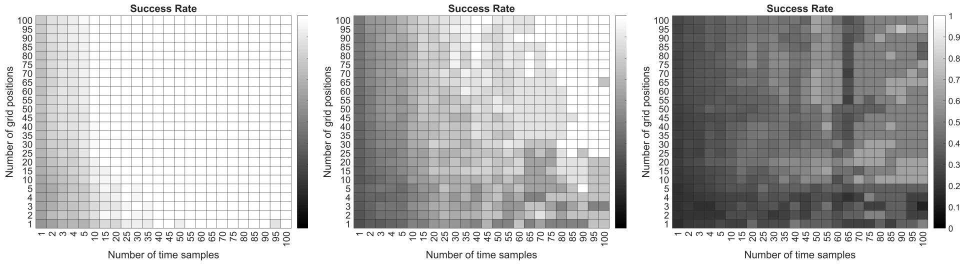
Having established a theoretical model for spatial leakages, we move towards a practical scenario. In particular, we exploit the available *location leakages* in the ARM Cortex-M4 so as to infer the accessed memory position of an 256-byte, data-independent LUT. Note that in the real chip we cannot isolate spatial from address leakage, i.e. we observe location leakage in its entirety. We use the template attack [6], i.e. we model the leakage using a multivariate normal distribution and attack trying to identify the key, or in our case region r of the SRAM.

The leakage vector $(\mathbf{L}|R = r)$ exhibits particularly large dimensionality and can generate a sizeable dataset, even for modest values of the grid dimension g. Thus, we employ dimensionality reduction techniques based on the correlation heuristic so as to detect points of interest (POIs) in the 300 × 300 grid and use a train-test ratio of 70-30. In addition, when performing template matching, we combine several time samples from the test set together (multi-sample/multi-shot attack), in order to reduce the noise and improve our detection capabilities[6]. We also opt for the improved template formulas by Choudary et al. [9] with pooled covariance matrix and numerical speedups. The goal of our template-based evaluation is not only to answer whether location exploitation is possible but also to gauge the effect of the experimental parameters ϵ on the exploitation process. Thus, similarly to Sections 3.2.1, 3.2.2, 3.2.5, we will investigate the effect of region partition, grid dimension and region placement in the real-world scenario. Unfortunately Sections 3.2.3 and 3.2.4 would require control over the manufacturing process (i.e. chips of different feature size) or control over regions with algorithmic noise (i.e. parallel memory activation), thus they cannot be tested in our current context. Throughout this section we will engage in comparisons between the theoretical model of Section 2 and our real-world attack, i.e. we will put the model's assumptions to test, discover its limitations and obtain more insight into the source of location leakage.

4.1 Area and Number of Regions

To observe the effect of partitioning, we gradually split the 256 bytes of the LUT into classes and built the corresponding template for each class. We perform a template attack on 2, 4, 8 and 16 partitions (with 128, 64, 32 and 16 bytes each respectively), i.e. we gauge the distinguishing capability of the adversary, as the number of components increases and their respective areas decrease. The results are visible in Figure 9, which showcases how the number of grid positions (spatial POIs) and samples/shots per attack affects the success rate (SR). The adversary can achieve a success rate of 100% when distinguishing between 2 or 4 regions, assuming that he uses multiple samples in his attack. The success rate drops to 75% for 8 and 50% for 16 regions, an improvement compared to random guess SRs of 12.5% and 6.25% respectively. Although we are not able to reach successful byte-level classification, we can safely conclude that location-based attacks are definitely possible on small LUTs and they can reduce the security level of an LUT-based implementations, unless address randomization countermeasures are deployed. When performing *single-shot* attacks, the template strategy becomes less potent, achieving SR of 57%, 33%, 17% and 11% for 2, 4, 8 and 16 regions, i.e. only slightly better than a random guess. In order to compare the success rate of the real attack to the theoretical model, we compute

[6] Whether this constitutes an option depends on the situation. If any sort of randomization such as masking or re-keying is present in the device then the adversary is limited in the number of shots that he can combine.



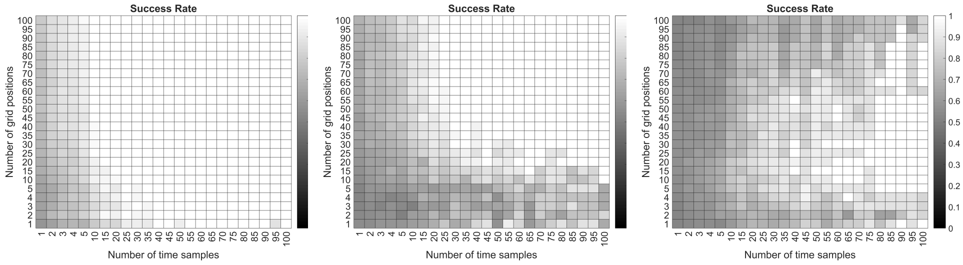
(a) 2 regions of 128 bytes each (b) 4 regions of 64 bytes each (c) 8 regions of 32 bytes each

Figure 9: The success rate of the template classifier as we partition the LUT. Y-axis denotes the number of spatial POIs used in model, X-axis denotes the number of samples used in attack. Scale denotes SR where white is 100% and black is 0%.

the model's SR for current device SNR under the same data complexity[7]. The model's *single-shot* SR is 99%, 50%, 13% and 12% for 2, 4, 8 and 16 regions respectively. We observe that the model follows the same trend, yet the device leakage exhibits divergences that indicate modeling imperfections.

4.2 Measurement Grid Dimension

Using the same approach, we evaluate the effect of grid dimension on the success rate of the template attack. We commence with the full 300 × 300 grid (2-day experiment) and subsequently scale down to 40×40 grid (1-hour) and 10×10 grid (2-minutes), as shown in Figure 10. We observe that for small grid sizes such as 10×10 the reduced dataset makes training harder, yet the multi-shot template attack is able to distinguish with SR equal to 100%. On the contrary, the theoretical model is unable classify correctly because the spatial POIs are often missed by such a coarse grid. To pinpoint this model limitation, we assess the spread of the POIs across the die surface and we visualize the best (according to correlation) grid positions in Figure 11. Interestingly, we discover numerous surface positions that leak location information, while being far away from the SRAM circuitry itself. This finding is in accordance with Unterstein et al. [51] on FPGAs. The figure suggests that location leakage is a combination of SRAM spatial leakage (as in the model), address leakage in the control logic and potentially out-of-model effects.



(a) 300 × 300 grid (b) 40 × 40 grid (c) 10 × 10 grid

Figure 10: The success rate of the 2-region template classifier as we decrease the experiment's grid size.

[7] The template attack uses the experimental data, while the theoretical SR uses simulated data of the same size and dimensionality.

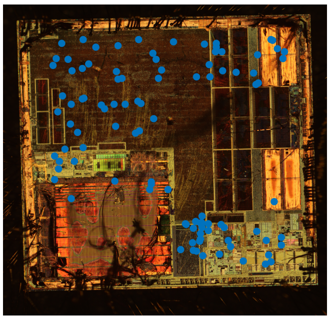
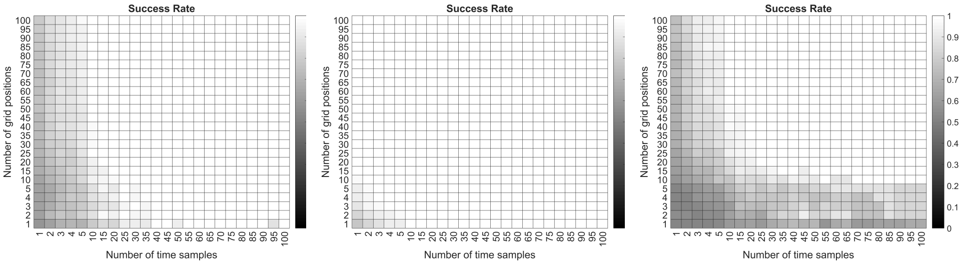


Figure 11: Spread of spatial POIs on chip surface.

4.3 Region Proximity and Interleaving

Finally, we evaluate the effect of region proximity and interleaving on the SR of templates. We examine close placement (adjacent SRAM regions), distant placement (SRAM regions at a large distance[8]) and word-interleaved placement (checkered SRAM regions). The results are visible in Figure 12. We observe that in all cases we reach multi-sample SR of 100%, in accordance with the theoretical model at the device SNR. However, attacking the word-interleaved LUT requires a bigger effort in modeling in terms of both grid POIs and samples per attack. Likewise, distinguishing between distant regions puts considerably less strain on the model. Thus, we conclude that distance and interleaving does indeed function like a countermeasure against location leakage, albeit it offers only mild protection in our ARM device.



(a) close (b) distant (c) word-interleaved

Figure 12: The success rate of the 2-region template-based classifier as we change the placement of regions.

5 Exploitation Using Neural Networks

Despite the fact that the multivariate normal leakage assumption is fairly realistic in the side channel context, applying distribution-agnostic techniques appears to be another rational approach [26]. Over the past few years, there has been a resurgence of interest in Deep Learning (DL) techniques, powered by the rapid hardware evolution and the need for rigorous SCA modeling [5, 29, 28, 27, 55, 37].

In this section, we evaluate the performance of various DL methods, including convolutional neural networks (CNNs, Subsection 5.1) and multi layer perceptrons (MLPs,

[8] Without knowledge of the chip layout we cannot be fully certain about the distance between memory addresses. Here we assume that the low addresses of the SRAM are sufficiently distant from mid ones, which are approx. 8 KBytes away.

Subsection 5.2), in inferring the activated region of the 256-byte, data-independent LUT on the ARM Cortex-M4. First, motivated by reusable neural networks, we use trained CNNs with all grid positions (no POI selection), but the results are fairly unsuccessful. We suspect that the results are affected by noise coming from spatial points that contain no location information. This approach can be compared to using full pictures for DL training.

To solve the above problem, we apply a dimensionality reduction based spatial POI selection similarly to Section 4.1. We notice that even a simple CNN already provides good results in a limited number of epochs and there is no gain in using a complex CNN. Thus, we move our attention to simpler MLPs. Interestingly, this approach surpasses the template attacks in effectiveness, enabling stronger location-based attacks that use less attack samples and can distinguish between smaller regions. The above method can be compared to using only the most meaningful parts of pictures for DL training.

5.1 Convolutional Neural Network Analysis

Fully pretrained CNNs. Before developing and customizing our own CNN model, we evaluate the performance of existing, state-of-the-art pre-trained networks. Pre-trained models are usually large networks that have been trained for several weeks over vast image datasets. As a result, their first layers tend to learn very good, generic discriminative features. Transfer Learning [34] is a set of techniques that, given such a pre-trained network, repurposes its last few layers for another similar (but not necessarily identical) task. Indeed, the objectives of our spacial identification task appear to be very close to those of standard image classification. Moreover, as outlined in Section 2, our data is formulated as 300×300 grid images, which makes them compatible with the input format of several computer vision classification networks. For this first attempt at CNN classification we use several state-of-the-art networks, namely Oxford VGG16 and VGG19 [42], Microsoft ResNet50 [15], Google InceptionV3 [50] and Google InceptionResNetV2 [49]. It should be noted that the input format of these networks is often RGB images, while our 300×300 heatmaps resemble single-channel, grayscale images. To address this and recreate the three color channels that the original networks were trained for, we experiment with two techniques; (1) we assemble triplets of randomly chosen heatmaps, and (2) we recreate the three color channels by replicating the heatmaps of the samples three times.

We apply the pretrained CNN classification on 2 closely placed SRAM regions of 128 bytes each. In accordance with the standard transfer learning methodology, during re-training we freeze the first few layers of the networks to preserve the generic features they represent. In each re-training cycle, we perform several thousand training-testing iterations. Despite all these and multiple hours of training, none of the aforementioned CNNs results into a retrained network with high classification success rate.

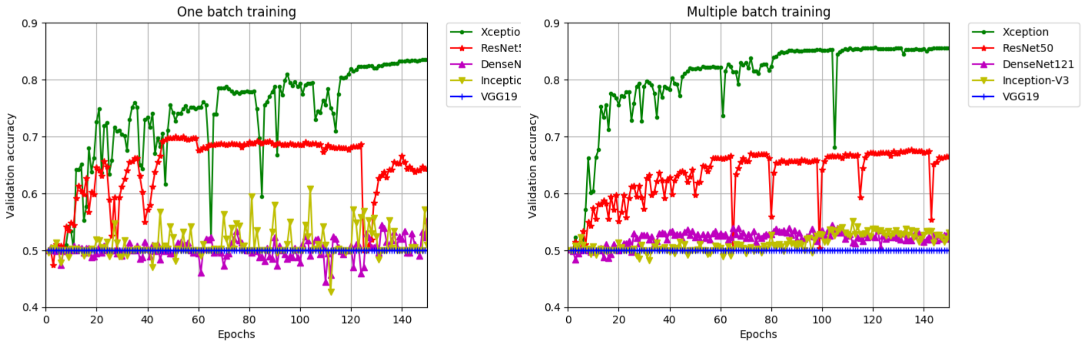
Custom pretrained CNNs. As a result of the low success rate of fully pretrained networks, we choose to proceed with Xavier [14] weight initialization and training from scratch. We observe that, despite the transformation of the sequential problem (SRAM accesses over time) to a spatial one, our dataset is dissimilar to visual classification datasets. Rephrasing, the images that we have to cope with feature intricate characteristics having little resemblance with those of the datasets that the pretrained CNN versions have been trained on, such as the ImageNet dataset [11]. Moreover, due to the fully distribution-agnostic approach, any randomly initialized CNN may suffer the effect of vanishing or exploding gradients, a danger that Xavier initialization should eliminate. The framework that was used for training and evaluating our customized CNNs is Keras [7] over TensorFlow [1] backend and the customized CNNs tested were VGG19 [42], InceptionV3 [50], ResNet50 [15], DenseNet121 [19] and Xception [8]. We also made use of the *scikit-learn* Python library [36] for the preprocessing of our data. The execution of this customized CNN training and testing was carried out in ARIS GRNET HPC (High-Performance Computing) infrastructure[9].

To gauge the effect of SRAM memory addressing on the training, all five CNNs are trained in two ways: one-batch training and multiple-batch training. During one-batch training we use location leakage from a single SRAM LUT, while for multiple-batch training we use four LUTs placed within a 16 KByte SRAM address range. The dataset is split into training, validation and test sets using a 70-20-10 ratio and is standardized by removing the median and scaling the data according to the quantile range. The networks are trained for 150 epochs of 32 images each, using the Adam optimizer [23] with default parameters. The results are visible in Figure 13.

We observe that the single-shot success rate of the Xception network (green line) exceeds by far all others' at 84% and the SR improves in stability when using multiple-batch training. It is worth noting that some CNNs, especially VGG19, remain incapable of learning anything meaningful about the discrimination of the two 128-byte regions. Another troubling fact is the sudden drops of validation accuracy during training time for both best-performing networks, Xception and ResNet50, a phenomenon rather indicative of overfitting. In our efforts to squeeze the best possible performance without sacrificing training stability and generalization capacity, we investigated the tolerance of the best performing network against two additional preprocessing techniques, namely sample-wise standardization and feature-wise standardization. The test set success rate of the three alternative techniques is visible in Table 2. Comparing with Section 4, we observe that CNNs are capable of surpassing the single-shot accuracy of template attacks, reaching 88% and making the CNN-based attack particularly useful against randomization countermeasures that limit the number of samples we can combine. Moreover we observe that spreading the training phase over several SRAM addresses (multiple batch) can assist classification, showing that the knowledge learned in a certain address range may be applicable elsewhere in the SRAM.

Table 2: Success rate of Xception network for alternative preprocessing techniques.

Alternative Pipeline	**Preprocessing step**	**Success Rate**
Xception-V1	dataset-wise, robust to outliers standardization	84.47 %
Xception-V2	sample-wise standardization	88.636 %
Xception-V3	feature-wise standardization	84.848 %



(a) Single-batch training.

(b) Multiple-batch training.

Figure 13: CNN validation accuracy for single/multiple-batch training.

[9] https://hpc.grnet.gr/en/

Table 3: Hyper-Parameters for training and validation.

Epochs	$30 - 80$ (depends on the number of regions)
Mini-Batch	100
Learning Rate	0.003
Learning Rate Decay Rate	0.5%
Learning Rate Decay Interval	100 epochs
L1	0.001
L2 (weight decay)	0.001
Weight Initialization	RELU
Activation Output Layer	SOFTMAX
Loss Function	NEGATIVELOGLIKELIHOOD
Updater	NESTEROVS
1 Dense Layer:	
- Number of Neurons:	20
- Activation Dense Layer:	TANH

5.2 Multi Layer Perceptron Network Analysis

We believe that the results from the previous section are affected by noise coming from spatial points that contain no location information. To eliminate this issue, we apply a dimensionality reduction techniques based on the correlation heuristic to detect the best spatial POI in the 300 × 300 grid, like in Section 4.1 Initial results using CNN show that even a simple CNN already provides good results in a limited number of epochs . Therefore there is no gain in using a complex CNN and we move our focus to simpler MLPs.

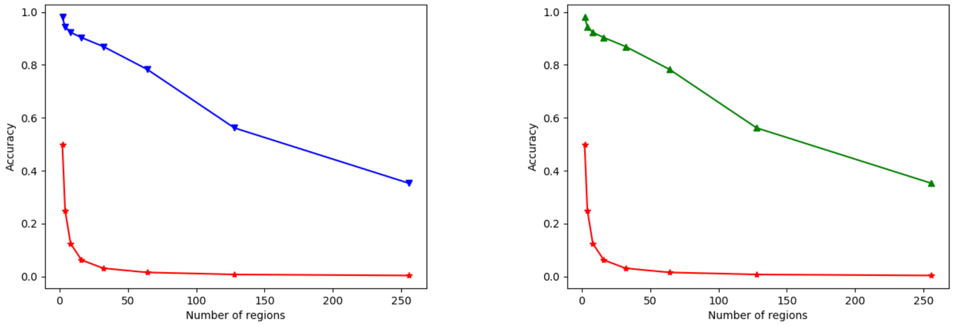
In [27], the authors presented how to use Multi Layer Perceptron (MLP) network to perform SCA on AES. In this section we present how to use MLP to recognize accesses to different addresses in the memory. Based on experiments, we discover that 5000 POI yielded the best network training.

We define our MLP to contain a single dense layer and used the back-propagation with NESTEROVS updater, with momentum 0.9, during training. The weights are initialized at random and applied to a RELU activation. The MLP is also configured with L1 and L2 regularization in order to improve the generalization. The analysis presented in this section is performed using Deep Learning for Java[10] in conjunction with Riscure Inspector deep learning functionality. To observe the effectiveness of MLPs, we gradually partition the 256 bytes of the LUT into classes and built the corresponding MLP for each class. We perform an MLP analysis on 2, 4, 8, 16, 32, 64, 128, and 256 partitions (with 128, 64, 32, 16, 8, 4, 2, and 1 bytes each, respectively). The dataset is split into training, validation and test sets using a 40-30-30 ratio. Then we select best hyper-parameters for training and validation[11] of our MLP network using a trial and error method. The chosen parameters are listed in Table 3.

The validation accuracy for 2, 4, 8, 16, 32, 64, 128, and 256 partitions is visible in Figure 14a. We have discovered that we achieve the best results for various numbers of epochs depending on the number of partitions. We have used 30 epochs for the 2 and 4 partitions, 40 epochs for the 16 and 32 partitions, 40 epochs for the 8 partitions, 70 for the 128 partitions, and 80 for the 64 and 256 partitions. Figure 14a indicates that the MLP network reaches high accuracy even for a large number of regions. To visualize the validation set success rate we present the validation final partitioning (for 16 partitions) in Appendix B. The greatest values are located on the diagonal and this indicates that the MLP learns correctly with high probability. The attack success rates for the test traces for 2, 4, 8, 16, 32, 64, 128, and 256 partitions are presented in Figure 14b; the exact accuracy values are 96%, 91%, 90%, 88%, 83%, 75%, 57%, and 32%, respectively. As expected, these values are slightly lower for the attacking phase than the validation ones in the

[10] https://deeplearning4j.org/

[11] The MLP parameters are chosen to maximize the attack success rate (which is equivalent to accuracy).



(a) Blue line denotes validation accuracy and red line denotes random guess success rate.

(b) Green line denotes attack success rate and red line denotes random guess success rate.

Figure 14: Validation accuracy for training and success rate for testing in MLP.

learning phase. We observe that even the SR for the 256 partitions, namely the 32% SR is significantly higher then a SR of a random guess: $1/256 = 4\%$.

Observing these results, we conclude that the MLP network can be substantially stronger than the template attacks when exploiting location leakage. It can achieve high SR using single-shot attacks, reaching 98% for 2 regions and 32% even when targeting single bytes in the SRAM. Notably, the MLP classification can strongly enhance the SR of a microprobe setup making it almost on par with the substantially more expensive photonic emission setup.

6 Conclusions & Future Directions

In this work, we have revisited the potent, yet often overlooked location-based leakage. We take the first steps towards theoretical modeling of such effects and we put forward a simple spatial model to capture them. Continuing, we demonstrate successful location-based attacks on a modern ARM Cortex-M4 using both standard template attacks, CNNs and MLPs. Throughout these attacks we assess the impact of various experimental parameters in order to elucidate the nature and exploitability of location-based leakage.

Regarding future work, we note that during the last years of side-channel research, the community has established a multitude of potent tools (ranging from Bayesian techniques to neural networks), all of which are particularly good at extracting the available leakage. Still, we remain far less capable of finding the exact cause behind it, especially in complex modern chips [4, 35]. Thus, a natural extension to this work is to delve deeper into the electrical layer of a system-on-chip, try to identify the “culprits” behind location leakage and ultimately diminish the emitted information. In the same spirit, we should strive towards improved circuit modeling, similarly to the works of Šijačić et al. [41] and Kumar et al. [25], adapt them to the spatial model and use it in order to shorten the development-testing cycle of products. Finally, going back to algorithmic countermeasures, we can start designing masking schemes that account for data *and* location leakage in order to provide a fully-fledged security that encapsulates multiple side-channel vulnerabilities.

Acknowledgments

We would like to thank Riscure BV, Rafael Boix Carpi, Ilya Kizhvatov and Tin Soerdien for supporting the process of chip decapsulation and scan.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Christos Andrikos, Giorgos Rassias, Liran Lerman, Kostas Papagiannopoulos, and Lejla Batina. Location-based leakages: New directions in modeling and exploiting. In *2017 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, pages 246–252, July 2017.

[3] Cédric Archambeau, Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. Template attacks in principal subspaces. In *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, pages 1–14, 2006.

[4] Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. In *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, pages 64–81, 2014.

[5] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 45–68. Springer, 2017.

[6] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, pages 13–28, 2002.

[7] François Chollet et al. Keras. `https://keras.io`, 2015.

[8] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016.

[9] Omar Choudary and Markus G. Kuhn. Efficient template attacks. In *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, pages 253–270, 2013.

[10] Thomas De Cnudde, Begül Bilgin, Benedikt Gierlichs, Ventzislav Nikov, Svetla Nikova, and Vincent Rijmen. Does coupling affect the security of masked implementations? In *Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers*, pages 1–18, 2017.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[12] Julien Doget, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert. Univariate side channel attacks and leakage modeling. *J. Cryptographic Engineering*, 1(2):123–144, 2011.

[13] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In Çetin K. Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2001*, pages 251–261, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[14] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *JMLR W&CP: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, volume 9, pages 249–256, May 2010.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] Wei He, Eduardo de la Torre, and Teresa Riesgo. An interleaved epe-immune pa-dpl structure for resisting concentrated em side channel attacks on fpga implementation. In Werner Schindler and Sorin A. Huss, editors, *Constructive Side-Channel Analysis and Secure Design*, pages 39–53, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[17] Johann Heyszl. Impact of Localized Electromagnetic Field Measurements on Implementations of Asymmetric Cryptography. https://mediatum.ub.tum.de/doc/1129375/1129375.pdf.

[18] Johann Heyszl, Stefan Mangard, Benedikt Heinz, Frederic Stumpf, and Georg Sigl. Localized electromagnetic analysis of cryptographic implementations. In *Topics in Cryptology - CT-RSA 2012 - The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, February 27 - March 2, 2012. Proceedings*, pages 231–244, 2012.

[19] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[20] Vincent Immler, Robert Specht, and Florian Unterstein. Your rails cannot hide from localized EM: how dual-rail logic fails on fpgas. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 403–424, 2017.

[21] Kouichi Itoh, Tetsuya Izu, and Masahiko Takenaka. Address-bit differential power analysis of cryptographic schemes ok-ecdh and ok-ecdsa. In *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '02, pages 129–143, London, UK, UK, 2003. Springer-Verlag.

[22] Dina Kamel, François-Xavier Standaert, and Denis Flandre. Scaling trends of the AES s-box low power consumption in 130 and 65 nm CMOS technology nodes. In *International Symposium on Circuits and Systems (ISCAS 2009), 24-17 May 2009, Taipei, Taiwan*, pages 1385–1388, 2009.

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[24] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, pages 388–397, 1999.

[25] A. Kumar, C. Scarborough, A. Yilmaz, and M. Orshansky. Efficient simulation of em side-channel attack resilience. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 123–130, Nov 2017.

[26] Liran Lerman, Romain Poussier, Olivier Markowitch, and François-Xavier Standaert. Template attacks versus machine learning revisited and the curse of dimensionality in side-channel analysis: extended version. *Journal of Cryptographic Engineering*, 8(4):301–313, Nov 2018.

[27] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 3–26. Springer, 2016.

[28] Zdenek Martinasek, Jan Hajny, and Lukas Malina. Optimization of power analysis using neural network. In *International Conference on Smart Card Research and Advanced Applications*, pages 94–107. Springer, 2013.

[29] Zdenek Martinasek and Vaclav Zeman. Innovative method of the power analysis. *Radioengineering*, 22(2):586–594, 2013.

[30] Philippe Maurine. *Securing SoCs in advanced technologies, https://cosade.telecom-paristech.fr/presentations/invited2.pdf.*

[31] M. Nassar, Y. Souissi, S. Guilley, and J. L. Danger. Rsm: A small and fast countermeasure for aes, secure against 1st and 2nd-order zero-offset scas. In *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1173–1178, March 2012.

[32] Kashif Nawaz, Dinal Kamel, François-Xavier Standaert, and Denis Flandre. Scaling trends for dual-rail logic styles against side-channel attacks: A case-study. In *Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers*, pages 19–33, 2017.

[33] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In *Information and Communications Security, 8th International Conference, ICICS 2006, Raleigh, NC, USA, December 4-7, 2006, Proceedings*, pages 529–545, 2006.

[34] Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[35] Kostas Papagiannopoulos and Nikita Veshchikov. Mind the gap: Towards secure 1st-order masking in software. In *Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers*, pages 282–297, 2017.

[36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[37] Emmanuel Prouff, Remi Strullu, Ryad Benadjila, Eleonora Cagli, and Cécile Dumas. Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. *IACR Cryptology ePrint Archive*, 2018:53, 2018.

[38] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A formal study of power variability issues and side-channel attacks for nanoscale devices. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, pages 109–128, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[39] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of aes. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, pages 413–427, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[40] Alexander Schlösser, Dmitry Nedospasov, Juliane Krämer, Susanna Orlic, and Jean-Pierre Seifert. *Simple Photonic Emission Analysis of AES*, pages 41–57. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[41] Danilo Sijacic, Josep Balasch, Bohan Yang, Santosh Ghosh, and Ingrid Verbauwhede. Towards efficient and automated side channel evaluations at design time. In Lejla Batina, Ulrich K\"uhne, and Nele Mentens, editors, *PROOFS 2018. 7th International Workshop on Security Proofs for Embedded Systems*, volume 7 of *Kalpa Publications in Computing*, pages 16–31. EasyChair, 2018.

[42] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[43] R. Specht, V. Immler, F. Unterstein, J. Heyszl, and G. Sig. Dividing the threshold: Multi-probe localized em analysis on threshold implementations. In *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 33–40, April 2018.

[44] Robert Specht, Johann Heyszl, Martin Kleinsteuber, and Georg Sigl. Improving non-profiled attacks on exponentiations based on clustering and extracting leakage from multi-channel high-resolution EM measurements. In *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers*, pages 3–19, 2015.

[45] Robert Specht, Johann Heyszl, and Georg Sigl. Investigating measurement methods for high-resolution electromagnetic field side-channel analysis. In *2014 International Symposium on Integrated Circuits (ISIC), Singapore, December 10-12, 2014*, pages 21–24, 2014.

[46] François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, pages 443–461, 2009.

[47] François-Xavier Standaert, Eric Peeters, Cédric Archambeau, and Jean-Jacques Quisquater. Towards security limits in side-channel attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, pages 30–45, 2006.

[48] Takeshi Sugawara, Daisuke Suzuki, Minoru Saeki, Mitsuru Shiozaki, and Takeshi Fujino. On measurable side-channel leaks inside ASIC design primitives. *J. Cryptographic Engineering*, 4(1):59–73, 2014.

[49] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.

[50] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[51] Florian Unterstein, Johann Heyszl, Fabrizio De Santis, and Robert Specht. Dissecting leakage resilient prfs with multivariate localized EM attacks - A practical security evaluation on FPGA. *COSADE*, 2017:272, 2017.

[52] Florian Unterstein, Johann Heyszl, Fabrizio De Santis, Robert Specht, and Georg Sigl. High-resolution em attacks against leakage-resilient prfs explained - and an improved construction. Cryptology ePrint Archive, Report 2018/055, 2018. https://eprint.iacr.org/2018/055.

[53] Harry L. Van Trees. *Detection, Estimation, and Modulation Theory: Part IV: Optimum Array Processing.* John Wiley and Sons, Inc., 2002.

[54] Neil Weste and David Harris. *CMOS VLSI Design: A Circuits and Systems Perspective.* Addison-Wesley Publishing Company, USA, 4th edition, 2010.

[55] Shuguo Yang, Yongbin Zhou, Jiye Liu, and Danyang Chen. Back propagation neural network based leakage characterization for practical security analysis of cryptographic implementations. In *International Conference on Information Security and Cryptology*, pages 169–185. Springer, 2011.

7 Appendix A

Algorithmic Noise in Tightly-Packed Surfaces. Since countermeasure designers opt often for algorithmic noise countermeasures, we investigate the statistical variance of N^{algo} for a tightly packed circuit that contains a large number of randomly switching components which try to hide the targeted component. We assume every noise-generating component to have area $b_i \approx d$, where d is the area of the targeted component Since we assume large n_a, both the noise-generating components as well as the targeted component are small w.r.t. the probe size, i.e. $d \ll o$. In a tightly packed circuit, the probe area o contains roughly $\frac{o}{d}$ randomly switching components, i.e. $n_a \approx \frac{o}{d}$. In this particular scenario, the following formula approximates N^{algo}.

$$N^{algo} = \sum_{i=1}^{n_a} N_i^{algo} = d \cdot \sum_{i=1}^{n_a} B_i = d \cdot A,$$

$$B_i \sim Bern(0.5) \text{ , } A \sim Binomial(n_a, 0.5)$$

$$\text{Thus, } N^{algo} \xrightarrow[\text{Theorem}]{\text{Central Limit}} Norm(\frac{d \cdot n_a}{2}, \frac{d \cdot n_a}{4})$$

Using the approximation of the Central Limit Theorem, we see that $Var[N^{algo}] = \frac{d \cdot n_a}{4} = \frac{o}{4}$. Thus, for the tightly-packed, small-component scenario we have established a direct link between the probe area o and the level of algorithmic noise, demonstrating how increasing the probe area induces extra noise.

8 Appendix B

Predicted versus actual values, visualizing the validation set success rate.

Predicted:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Actual:																
0	35	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
1	1	30	1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	40	1	1	0	0	0	0	0	0	0	0	0	0
4	2	0	0	1	31	0	0	1	0	0	0	0	1	0	0	0
5	0	1	0	1	0	28	1	0	0	0	0	0	0	0	0	0
6	0	1	0	0	0	0	37	0	0	0	0	0	0	1	0	0
7	0	0	0	1	2	1	0	26	0	0	0	0	0	0	1	0
8	0	0	0	1	0	0	0	0	26	0	0	0	0	0	0	0
9	0	0	1	0	0	1	1	0	1	30	0	0	0	0	0	0
10	0	0	0	0	0	2	0	0	1	1	37	0	1	0	0	2
11	0	1	0	1	1	0	0	1	0	1	0	34	0	1	0	0
12	0	1	0	0	0	0	0	2	1	0	0	0	34	0	0	0
13	0	0	0	0	0	0	1	1	0	0	3	0	0	32	2	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0
15	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	31